PROCEEDINGS OF SCIENCE

Integrating AI into Data-Center Operations: Methods for Detection, Prediction, and Adaptive Control

Marco Lorusso,[a,*] **Daniele Bonacorsi,**[b,c] **Luca Dell'Agnello,**[a] **Diego Michelotto**[a] **and Lucia Morganti**[a]

[a] *CNAF Division, Istituto Nazionale di Fisica Nucleare, Viale Berti Pichat 6/2, Bologna, Italy*

[b] *Bologna Division, Istituto Nazionale di Fisica Nucleare, Viale Berti Pichat 6/2, Bologna, Italy*

[c] *Alma Mater Studiorum - University of Bologna Viale Berti Pichat 6/2, Bologna, Italy*

E-mail: marco.lorusso@cnaf.infn.it

The increasing complexity and scale of modern data centers generate operational environments where the ability to detect anomalies, anticipate failures, and optimize resource usage is becoming critically important. Recent advances in machine learning and artificial intelligence offer powerful techniques for extracting actionable insights from heterogeneous monitoring data, ranging from logs and metrics to event streams and security signals. In this contribution, we explore an AI-driven approach that can be applied to a specific large-scale computing facility hosted at INFN-CNAF in Bologna, as a precursor for methods for anomaly detection, predictive alerting and fault classification. This study can establish the methodological foundations for future integration with the INFN-CNAF Big Data Platform, which is expected to provide the unified data backbone for advanced operational analytics. The goal of this contribution is to highlight promising research directions and practical use cases where AI can provide measurable value to large distributed computing infrastructures.

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)004

1. Introduction

Managing and monitoring modern data centers have become increasingly challenging because of their growing scale, complexity, and hardware heterogeneity. Facilities supporting scientific computing must handle thousands of interconnected components, ranging from compute nodes and storage systems to complex networking infrastructures, especially when tasked to provide cloud-like services and large-scale and distributed data processing. In such environments, even minor inefficiencies or undetected anomalies can propagate quickly, leading to performance degradation or service disruption.

In this context, the Computing division of the Italian National Institute for Nuclear and Subnuclear Physics (INFN), called CNAF [1], has to keep up with this trend employing new technologies like AI and predictive maintenance. Indeed, CNAF, and the data centre it manages in Bologna (Italy), plays a central role in supporting large-scale scientific computing, most notably as a Tier-1 facility of the CERN Worldwide LHC Computing Grid (WLCG). Consequently, the facility must manage substantial data volumes while providing reliable, high-throughput processing capabilities. At the same time, CNAF is involved in a broader ecosystem that includes technology transfer activities and the coordination of distributed cloud infrastructures, further increasing the diversity of services and operational requirements.

This complexity has motivated a progressive consolidation of monitoring and management tools, with the adoption of an infrastructure based on systems such as Sensu [2], InfluxDB [3], and Grafana [4] to collect and visualise metrics across CNAF services. In parallel, a complementary stack based on Filebeat [5] and OpenSearch [6] is being developed to monitor logs coming from cloud resources, virtual machines, and applications within the INFN Cloud framework.

Despite these efforts, current monitoring practices remain largely centred on metrics and rely on manual analysis. The increasing scale and heterogeneity of the infrastructure make it difficult to fully exploit the available data, particularly when considering the integration of different data sources such as logs, metrics, and application-level information. In particular, the inclusion of logging data into the monitoring pipeline and the automation of data analysis represent ongoing challenges, currently addressed only at an exploratory stage.

That is why the need for more advanced approaches to operational intelligence is growing, capable of handling heterogeneous data sources and supporting automated decision-making processes. In this work, we investigate the adoption of an artificial intelligence model to test an approach for enhancing data-centre operations at CNAF. The focus is on enabling more effective detection of anomalous conditions, improving the ability to anticipate failures.

2. Dataset Description and Preprocessing

Several distinct data sources could contribute to the dataset produced by a data centre, including system metrics, monitoring outputs, and logs produced by hosts and services. These data streams differ not only in semantics but also in structure: metrics are typically represented as regular time-series, logs are semi-structured and event-driven, and monitoring data may include structured formats such as JSON. This heterogeneity introduces non-trivial challenges in terms of data integration, alignment, and interpretation.

PoS(ISGC2026)004

The dataset originates from the operational infrastructure managed by INFN-CNAF, although this initial study considers only numerical monitoring data, as a first stepping stone and proof of concept before further developments. Still, the data already reflects the intrinsic complexity of a production environment, where multiple subsystems generate high-volume telemetry streams: the CNAF facility operates thousands of compute jobs concurrently and manages large storage systems, resulting in continuously growing data volumes. Telemetry is collected from a large number of nodes and services, producing high-frequency measurements that must be processed and analysed efficiently. As a consequence, the dataset exhibits both high dimensionality and significant temporal depth, making conventional analysis techniques difficult to apply without prior transformation.

2.1 Data Preparation

The initial data consisted of multi-gigabyte JSON dumps characterised by a highly redundant structure. Direct use of this raw data is impractical for anomaly detection tasks, both due to its size and the presence of noise and inconsistencies.

A preprocessing pipeline was therefore designed to extract a consistent and informative representation of system behaviour, i.e. a high-quality multivariate time series, suitable for machine learning models.

Starting from the JSON records exported from an InfluxDB [3] time-series database, the multiple metrics per host and timestamp were parsed to reorganize the data into host-time-metric representation, while selecting the most representative system-level metrics (e.g. CPU usage, load averages, memory utilization, disk I/O) to focus on those with high diagnostic value. At the same time, sparse and noisy process-specific metrics were removed to reduce noise and dimensionality.

This process resulted in a structured dictionary where each host was associated with a set of timestamps, and each timestamp contained a vector of selected metrics, enabling consistent alignment across hosts and facilitating subsequent transformations. Indeed, this allowed the exclusion of timestamps not populated by all the hosts and eliminate hosts with not enough snapshots, due to their age, to create a completely coherent and dense dataset. For each valid timestamp, a matrix of shape hosts $\times$ metrics was generated (304×36), guaranteeing that the final dataset can be represented as

$$X \in \mathbb{R}^{T \times N \times F} \tag{1}$$

where T is the number of time steps, N the number of hosts, and F the number of features.

Given the heterogeneous scale of system metrics, feature-wise normalization was applied, computing scaling parameters exclusively on the training split of data to avoid data leakage. Thus, each feature was rescaled to a common range using:

$$X' = \frac{X - \min(X_{train})}{\max(X_{train}) - \min(X_{train})} \tag{2}$$

This step is essential for stabilizing the training of Neural Networks and ensuring comparability across features.

The processed dataset is then further transformed into overlapping temporal windows to capture short-term dynamics using a sliding window approach. Sequences of fixed length W equal to 6 were extracted with a stride of 1. Each sample therefore represents a tensor of shape:

$$(W \times N \times F) = (6 \times 304 \times 36) \tag{3}$$

PoS(ISGC2026)004

enabling the model to learn temporal correlations both within individual hosts and across the infrastructure.

2.2 Synthetic Anomaly Injection

The data used in this study is collected from production systems, where anomalous conditions are relatively rare and often not systematically labelled. This scarcity of ground truth poses a limitation for the evaluation of anomaly detection methods. To address this issue, synthetic anomalies are injected into a subset of the dataset used for validation. The injection process is designed to simulate realistic operational events affecting randomly selected nodes and metrics, while preserving the statistical properties of the original data.

The injection procedure operated by sampling a set of temporal segments over the dataset, defined by a start time, a duration, a subset of affected hosts and features, and an anomaly type between:

Transient spikes : short and sparse increase in CPU usage or disk I/O activity, simulating bursty workloads or sudden contention events;

Sustained workload shifts : persistent deviations from baseline behaviour over an entire time segment, reproducing long-running workload imbalances or configuration changes;

Gradual drifts (e.g. memory leaks): linearly increasing deviations over time, reproducing progressive degradation phenomena;

Noise bursts : high-variance stochastic perturbations affecting selected metrics, simulating unstable or noisy subsystems;

Stuck values : metrics frozen at a constant value, representing monitoring failures or stalled processes;

Host outages : complete suppression of all metrics for selected hosts, implemented as zeroing of feature values, to mimic node failures or disconnections;

Rack-level anomalies : simultaneous perturbations affecting groups of hosts, introducing spatial correlation due to shared infrastructure issues;

Pattern swaps : replacement of a host's behaviour with that of another host over a different time interval.

The number of anomalous segments was controlled as a fraction of the total timeline (e.g., 10–15%), ensuring that anomalous conditions remain sparse relative to normal behaviour. Segment lengths are sampled from a bounded interval to simulate both short-lived transients and longer-lasting degradations. For each segment, binary labels were also generated at the host-time level, marking all affected hosts during the anomaly duration, enabling the computation of evaluation metrics such as ROC curves, AUC, and F1-score.

In Figure 1 a comparison is shown between a representative time snapshot before and after anomaly injection. The perturbations are not visually distinguishable from normal variability, confirming that the injected anomalies are subtle and require data-driven methods to be detected.

PoS(ISGC2026)004

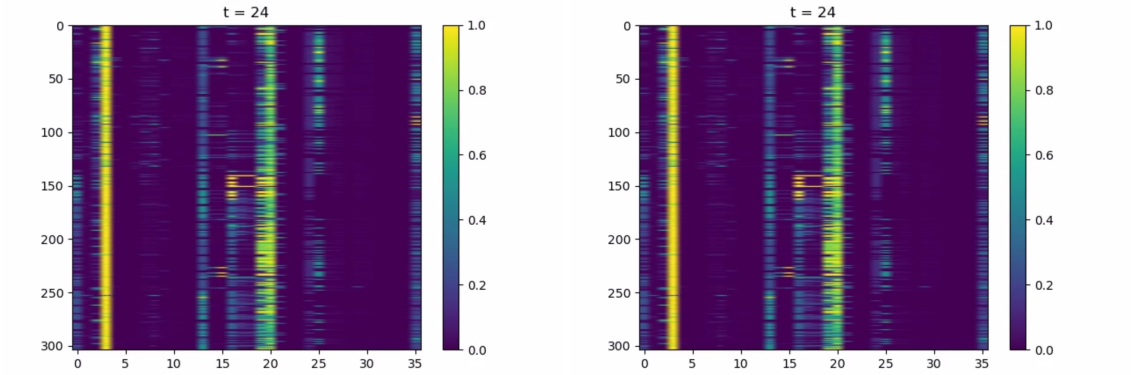

Figure 1: Heatmaps of a single time snapshot ($t = 24$) before (left) and after (right) anomaly injection. The perturbations are not visually distinguishable, highlighting their subtlety.

3. Neural Networks for Time-Series Anomaly Detection

The task introduced in the first section of this paper falls in the general category of Anomaly Detection (AD). AD [7] aims to identify instances containing patterns that deviate from those observed in normal instances [8]. This task is crucial in various vision applications, such as manufacturing defect detection, medical image analysis, and video surveillance. Unlike typical supervised classification problems, anomaly detection presents unique challenges. Primarily, it is difficult to obtain a substantial amount of anomalous data, whether labeled or unlabeled. Additionally, the differences between normal and anomalous patterns are often fine-grained, as defective areas can be small and subtle in high-resolution images. Since the distribution of anomaly patterns is unknown in advance, models are trained to learn the patterns of normal instances. In practice, an instance is determined to be anomalous if it is not well-represented by these models. Considering the peculiarities of this kind of task, Anomaly detection is therefore a common application of autoencoders. [9].

In literature there are a great amount of models, both inside and outside the realm of Machine Learning, built to tackle this problem. Here we will focus on a couple of Neural Networks suited for it, as a small set of tested alternatives for our use case, ending up with a more in-depth description of the "mini-transformer" architecture used for the final results presented in this work.

3.1 Autoencoders

An Autoencoder (AE) is a type of Neural Network that is trained to attempt to copy its input to its output [10]. Internally, it usually comprises of a hidden layer h that describes a code used to represent the input. The network can be viewed as made up of two parts: an encoder function $\mathbf{h} = f(\mathbf{x})$ and a decoder that produces a reconstruction $\mathbf{r} = g(\mathbf{h})$. This architecture is presented in Figure 2. If an Autoencoder succeeds in simply learning to set $g(f(\mathbf{x})) = \mathbf{x}$ everywhere, then it is not especially useful. Instead, Autoencoders are designed to be unable to learn to copy perfectly. Usually they are restricted in ways that allow them to copy only approximately, and to copy only input that resembles the training data. Because the model is forced to prioritize which aspects of the input should be copied, it often learns useful properties of the data.

PoS(ISGC2026)004

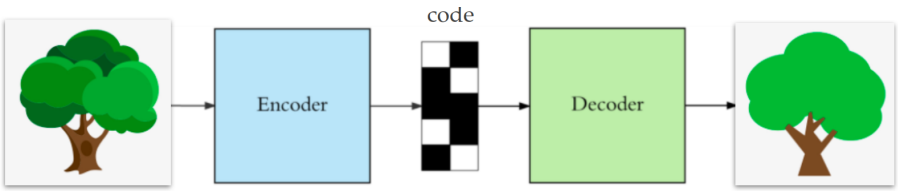

Figure 2: The general structure of an autoencoder, mapping an input to an output (called reconstruction) through an internal representation or code. The autoencoder has two components: the encoder and the decoder.

Autoencoders with nonlinear encoder functions f and non-linear decoder functions g can learn a more powerful non-linear generalization of Principal Component Analysis [11]. In other words, it is able to simplify the data while preserving its essential patterns and structures.

3.2 Recurrent Neural Networks

Recurrent neural networks (RNNs) [10], are a class of Neural Networks designed for processing sequential data. Instead of being specialized for processing a grid of values, like Convolutional Networks (CNN) for images, a RNN is a Neural Network that is built for processing a sequence of values $x^{(1)}, \ldots, x^{(\tau)}$, capturing temporal dependencies across elements. Just as CNNs can readily scale to images with large width and height, RNNs can scale to much longer sequences than would be practical for networks without sequence-based specialization. Most Recurrent Networks can also process sequences of variable length.

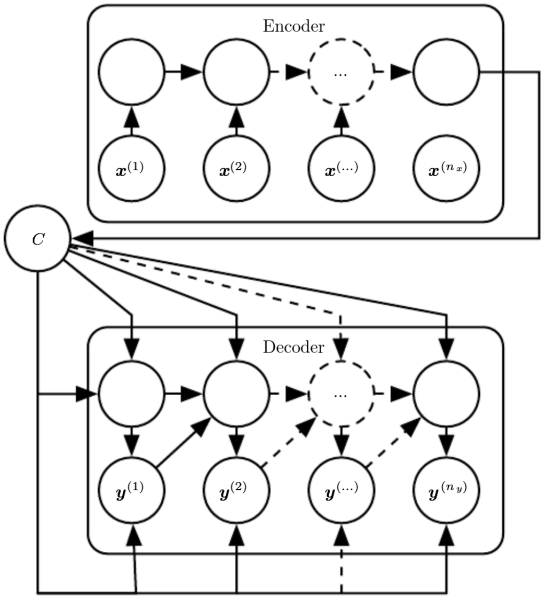

Figure 3: Encoder–decoder (seq-to-seq) RNN architecture. An encoder processes the input sequence and compresses it into a context vector, which is then used by a decoder to generate the output sequence.

RNNs can be extended to an encoder–decoder framework for sequence-to-sequence (seq-to-seq) learning, where an input sequence is mapped to an output sequence through an intermediate

PoS(ISGC2026)004

representation (see Figure 3). In this setting, an encoder RNN processes the input sequence and compresses it into a fixed-size context vector, which summarizes the relevant information. A decoder RNN then uses this representation to reconstruct the original sequence (in the case of an autoencoder) or to generate a new sequence. When applied in an AE configuration, this architecture is referred to as a sequence-to-sequence Autoencoder, where the model is trained to reconstruct the input sequence from its compressed representation. Similarly to standard AEs, the bottleneck imposed by the context vector forces the network to learn meaningful temporal features, capturing the underlying structure of the sequence rather than simply memorizing it.

3.3 Transformers

Recurrent neural networks, and in particular Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), have long represented the state of the art for sequence modelling and transduction tasks such as language modelling and machine translation. However, these models process sequences in a strictly sequential manner: the hidden state at time t, $h(t)$, depends on the previous state $h(t-1)$ and the current input.

Attention mechanisms were introduced to address this limitation by allowing models to directly relate different positions in a sequence, regardless of their distance. Instead of relying solely on sequential state updates, attention enables the model to focus on the most relevant parts of the input when computing each representation. Building on this idea, the Transformer architecture [12] removes recurrence entirely and relies exclusively on attention mechanisms to model dependencies across the sequence.

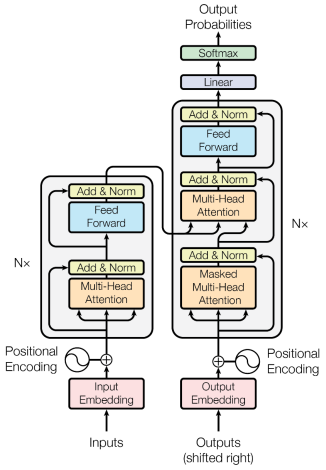

Figure 4: The Transformer model architecture from [12].

The Transformer follows an encoder–decoder structure similar to previous sequence-to-sequence models (shown in Figure 4). The encoder maps an input sequence $(x^{(1)}, \ldots, x^{(n)})$ into a sequence of continuous representations, while the decoder generates the output sequence $(y^{(1)}, \ldots, y^{(m)})$ in an autoregressive manner, conditioning each prediction on previously generated outputs. Both encoder and decoder are composed of stacked layers that operate on the entire sequence, allowing the model to build rich contextual representations.

PoS(ISGC2026)004

At the core of the Transformer lies the self-attention mechanism, which computes representations of a sequence by explicitly modelling interactions between all pairs of positions. Given a set of queries Q, keys K, and values V, attention is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4}$$

where $\sqrt{d_k}$ is a scaling factor to stabilize training. This expression produces a weighted combination of values where the weights reflect the relevance of each position with respect to the query. To increase the expressiveness of the model, Transformers employ Multi-head Attention, where multiple attention operations are performed in parallel on different learned projections of the input, each capturing different types of relationships within the sequence, enabling the model to jointly attend to information from multiple representation subspaces.

3.4 A mini-Transformer Autoencoder: TransAE

The Transformer is particularly well-suited for modelling multivariate time series data collected from distributed computing systems. Unlike recurrent approaches, self-attention enables the model to capture both temporal dependencies and correlations across different hosts and metrics without relying on sequential processing. This makes it sensitive to small deviations across multiple nodes other than isolated events. By leveraging attention mechanisms, the model can dynamically focus on the most informative features and time steps, facilitating the learning of complex system-wide patterns.

However, a full Transformer typically contains millions of parameters, making it less suitable for online inference or environments where it must operate alongside other computational workloads. This is why, for this work, it was decided to use only the encoder (see Figure 5) from [12] to keep the number of parameters as low as possible to achieve high portability and low inference times.

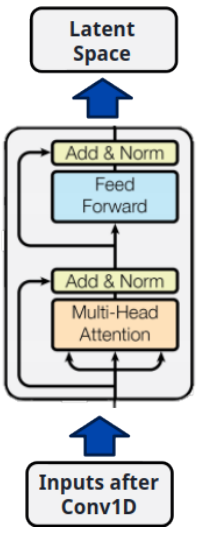

Figure 5: Encoder block of the Transformer model from [12].

The final architecture designed for the anomaly detection task, shown in Figure 6, is an Autoencoder composed of the following components

PoS(ISGC2026)004

Encoder :

- Per-host temporal Conv1D layers extract short-term patterns;
- A global max pooling operation produces a compact representation for each host;

Cross-host modelling :

- Host embeddings are processed through a Multi-head Attention mechanism to capture inter-host dependencies;
- Residual connections and feed-forward layers refine the contextual representations;
- A global latent vector is computed to summarize the overall system state;

Decoder :

- Host embeddings are expanded and passed through temporal Conv1D layers;
- The original telemetry window is reconstructed.

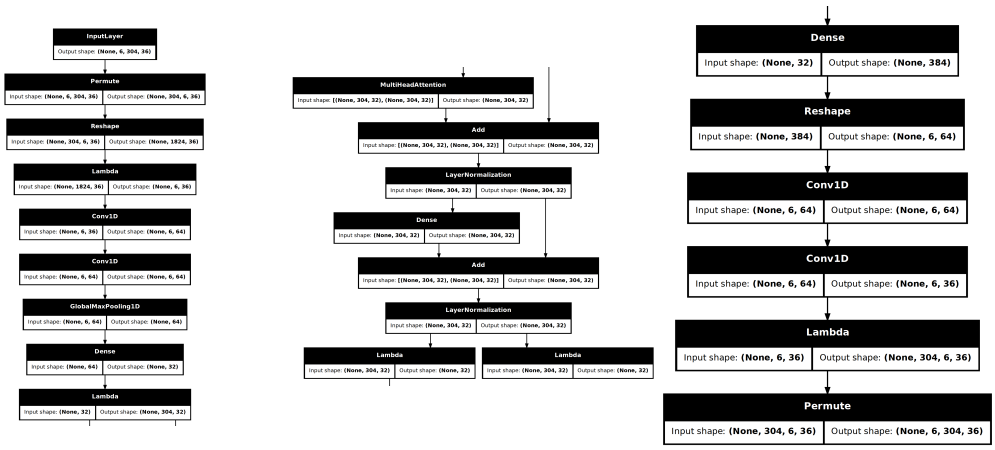

Figure 6: Diagram of the proposed Autoencoder architecture (TransAE), showing the encoder (left), cross-host attention module (centre), and decoder (right) components as implemented in Keras.

The resulting model, TransAE, comprises 87524 trainable parameters, placing it at the lightweight end of this class of architectures.

4. Anomaly Detection Results

The model described in the previous section was trained for 8 epochs and a fixed learning rate of 0.003 using time-series containing no injected anomalous signals (80% of whole dataset available), and tested against series with randomly injected signals, as described in Section 2.2. The anomaly score was computed from the reconstruction error of each host within the input time window, and then aggregated by averaging the errors of the top 5 most anomalous hosts to obtain a final score.

PoS(ISGC2026)004

In order to have a comparison with a different model, another Neural Network was designed based on the RNN architecture by replacing the Transformer block with a Recurrent Long-Short-Term Memory block distributed separately for each host in the data entry. The ROC Curve for both this baseline and the TransAE can be found in Figure 7, while a summary of some numerical values for True Positive Rates (TPR) at values of False Positive Rate (FPR) equal to 10% and 20% together with the Area Under the Curve of the ROC, are listed in Table 1.

	TPR @ FPR = 10%	**TPR @ FPR = 20%**	**AUC**
TransAE	0.84507042	0.92957746	96%
RVAE	0.61971831	0.66901408	79%

Table 1: Performance comparison between the proposed TransAE model and the RVAE baseline. The table reports True Positive Rate (TPR) at fixed False Positive Rate (FPR) thresholds (10% and 20%), along with the overall Area Under the ROC Curve (AUC).

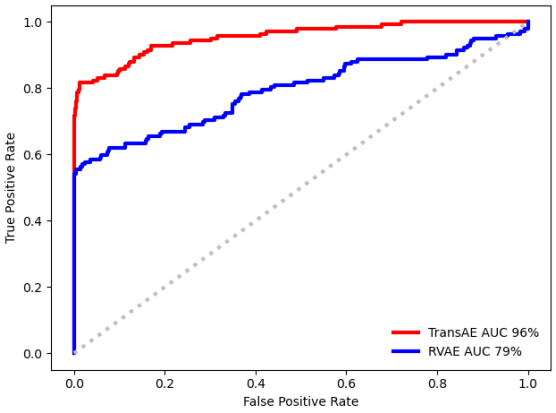

Figure 7: ROC curves comparing the proposed TransAE model (red) with the RVAE baseline (blue). The TransAE achieves a higher AUC (96%) compared to RVAE (79%), indicating improved anomaly detection performance across operating points.

From these results, a clear improvement can be observed when adopting a more advanced, context-aware encoding mechanism such as multi-head attention. This performance gain is particularly notable considering that the RVAE architecture requires the training of approximately 200.000 parameters, significantly more than the proposed model.

While ROC curves provide a threshold-independent evaluation of the model performance, selecting an operating point requires analysing the trade-off between precision and recall. To this end, the F1-score was evaluated as a function of the decision threshold, as shown in Figure 8.

The F1-score reaches its maximum for low threshold values, indicating that the model achieves the best balance between precision and recall in this region.

5. Conclusion

In this work, the use of deep learning techniques for anomaly detection in large-scale data-centre environments was investigated. In particular, a model based on the Autoencoder architecture

PoS(ISGC2026)004

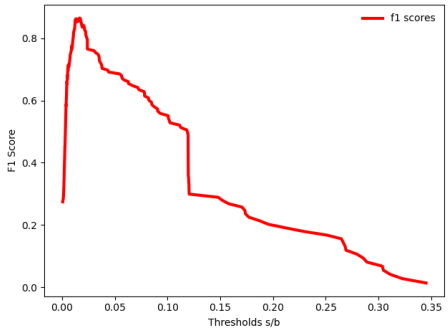

Figure 8: F1-score as a function of the decision threshold for the TransAE model. The curve highlights the trade-off between precision and recall and identifies the threshold region yielding optimal detection performance.

was tested as an effective unsupervised framework for identifying deviations from normal behaviour without requiring labelled data. This was possible by performing temporal feature extraction and cross-host modelling to better represent the underlying structure of the data-centre, with respect to more classical models based on Recurrent Neural Networks.

A key contribution of this work is the integration of Attention Mechanisms to explicitly model dependencies between hosts. This allowed the model to move beyond isolated node-level analysis and capture coordinated patterns across the infrastructure, which are often indicative of system-wide anomalies. Analysing the results show that this approach leads to improved detection performance compared to recurrent methods, while maintaining a relatively compact model size.

Despite these promising results, several aspects can be further improved. Future work will focus on enhancing model robustness through more advanced feature engineering and the incorporation of additional data sources. Moreover, extending the architecture to better capture both temporal dynamics and cross-host interactions remains an important direction, potentially through more expressive attention-based designs. Finally, the development of a dedicated data ingestion and preprocessing pipeline will be essential to enable systematic experimentation and support deployment in production environments.

References

[1] CNAF Website. https://www.cnaf.infn.it/en.

[2] Sensu Website. https://www.sensu.io.

[3] Influxdb Website. https://www.influxdata.com/.

[4] Grafana Website. https://grafana.com/.

[5] Filebeat Website. https://www.elastic.co/beats/filebeat.

[6] OpenSearch Website. https://opensearch.org/.

PoS(ISGC2026)004

[7] M. Lorusso, *Optimization of ML-Based BSM triggering with Knowledge Distillation for FPGA implementation in the CMS Level-1 Trigger*, Ph.D. thesis, Alma Mater Studiorum - University of Bologna, 2025.

[8] C.L. Li, K. Sohn, J. Yoon and T. Pfister, *Cutpaste: Self-supervised learning for anomaly detection and localization*, *ArXiv e-prints* (2021) [2104.04015].

[9] R. Chalapathy and S. Chawla, *Deep learning for anomaly detection: A survey*, *ArXiv e-prints* (2019) [1901.03407].

[10] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press (2016).

[11] I.T. Jolliffe and J. Cadima, *Principal component analysis: a review and recent developments*, *Philos. Trans. A Math. Phys. Eng. Sci.* **374** (2016) .

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez et al., *Attention is all you need*, *ArXiv e-prints* (2023) [1706.03762].

PoS(ISGC2026)004