Compression-Based Anomaly Detection for Business Intelligence: A Study of LZ78 and Neural Models in Time Series

David Constantin BERBECE
Bucharest University of Economic Studies, Bucharest, Romania
berbecedavid25@stud.ase.ro

Ioan David PASARIN
University of Bucharest, Bucharest, Romania
david-ioan.pasarin@s.unibuc.ro

Maria PREDA
University of Bucharest, Bucharest, Romania
maria.preda@s.unibuc.ro

Abstract. *Anomaly detection in time series is an important problem in domains where abnormal behavior may indicate critical system failures and labeled data is rarely available, making unsupervised approaches particularly relevant. This paper relies on two compression-based methods for time series anomaly detection. The first approach implements universal probability assignment derived from the Lempel-Ziv 78 compression algorithm, while the second method employs neural lossless compression using an autoregressive recurrent model. Both techniques interpret anomalies as sequences that are difficult to compress under models learned from normal behavior. The methods are evaluated on the Electricity Transformer dataset, focusing on the temporal behavior and statistical properties of the anomaly scores, in order to highlight the complementary characteristics of universal and learned compression strategies. By eliminating the need for costly manual labeling, these unsupervised models provide a scalable and cost-efficient solution for real-time monitoring in resource-constrained industrial environments. This research demonstrates how information-theoretic metrics can be effectively integrated into Business Intelligence systems to facilitate informed decisions regarding predictive maintenance and risk mitigation.*

Keywords: Anomaly detection, Business Intelligence, Time series, Data compression, LZ78 algorithm, Neural networks, Unsupervised learning.

1 Introduction

In the current era of Big Data, the primary objective of Business Intelligence is to extract actionable insights from vast temporal datasets, enabling managers to make informed decisions before critical system deviations escalate into economic losses.

First introduced in economics and finance to model temporal data such as stock prices and market indicators, time series analysis was later extended to a variety of fields, including energy systems, weather forecasting, healthcare systems, and industrial process control.

In time series data, current observations depend on past behavior, which is the main indicator for prediction. Thus, it is essential for the data to be organised temporally, even though the underlying distribution may evolve over time, forming seasonal patterns that repeat periodically, trends that capture long-term changes, and regimes that reflect different operational states of the system.

As they are employed in critical domains, the behavior of time series must be continuously monitored, so that emerging changes can be detected early, either to prevent potential

failures or to enable appropriate intervention, since such deviations may indicate abnormal situations with significant impact on the system.

For instance, in energy systems, transformers are continuously monitored through sensors to track the oil temperature. An abnormally high temperature increase may indicate several problems, including insulation degradation, which can eventually result in irreversible failures.

In modern industrial management, the ability to preemptively identify system deviations is directly linked to reducing operational risks and minimizing the substantial economic losses associated with unplanned downtime.

In this context, the need for anomaly detection arises.

1.1 Anomaly detection in time series

Anomaly detection in time series focuses on identifying observations and temporal patterns that deviate from the expected behavior of a system. Due to the strong temporal dependencies between consecutive data points, such deviations are often difficult to distinguish from normal variations, which may naturally arise from noise, seasonality, or evolving trends.

Anomalies can manifest in different forms, ranging from isolated abnormal points to short irregular subsequences or persistent changes in the operating regime of the system. In many practical scenarios, explicit anomaly labels are unavailable, which makes unsupervised approaches particularly suitable. These methods typically rely on learning a representation of normal behavior from historical data and evaluating how unlikely new observations appear under the learned model.

Time series anomaly detection has therefore become an essential component in applications such as energy systems, industrial monitoring, healthcare supervision, and network security, where the timely identification of abnormal behavior can support early warning mechanisms and help prevent critical system failures.

An anomaly in this domain can manifest in various forms, from a minor deviation from the normal behavior of a time series to a persistent change in its behavior over a long period of time. Since it is not realistic to account for all possible anomalies that can occur, a commonly adopted approach in many fields, including computer networks, is to learn the normal behavior of the time series at training, and, afterwards, to notice as fast as possible any changes that might appear.

2 Literature Review

Early approaches to time series anomaly detection relied on statistical models such as autoregressive or ARIMA-based methods, where anomalies are identified as large deviations from predicted values.

Machine learning techniques were used more recently, as autoencoders and recurrent neural networks, which learn representations of normal behavior, reconstruct it and flag the anomalies. They often require large amounts of data, and their performance may degrade in non-stationary environments.

Another direction is based on information-theoretic principles, where anomalies are defined as events that exhibit low probability or poor compressibility under a model trained on normal data. Compression-based methods provide a natural unsupervised framework and have been successfully applied in many domains. The approaches considered in this work follow this perspective, focusing on probabilistic modeling through lossless compression.

We mainly relied on two compression-based approaches: a neural lossless compression framework proposed in [1] and a universal anomaly detection method based on LZ78 compression introduced in [2]. Both use compression methods to estimate probabilities for unseen data

Algorithm 1: Building Dictionary using LZ78

Data: $adddabcs$
Result: $\{a, d, dd, ab, c, s\}$

Initialize dictionary $D = \emptyset$
Set current string $w = \epsilon$
foreach *symbol* x *in input sequence* **do**
    if $w \cdot x \in D$ **then**
        $w \leftarrow w \cdot x$
    else
        Add $w \cdot x$ to dictionary D
        Output $w \cdot x$
        $w \leftarrow \epsilon$
if $w \neq \epsilon$ **then**
    Add w to dictionary D

and, based on these values, compute anomaly scores for new cases. Although the first method is especially designed for time series behavior, the second one was implemented and tested on computer networks and attack identification, specifically botnets.

We compared the results of the two methods on the ETTh2 dataset, adapting the algorithms so that they match the studied topic.

3 LZ78 Approach

In the first approach, proposed in [2], the authors proposed an anomaly detection method based on LZ78 compression. This type of compression is almost optimal, achieving a compression rate close to the entropy of the source. In short, it constructs a rooted tree using sequences from the training data and assigns probabilities to the edges.

During the testing stage, new sequences are traced through the tree in a manner similar to the compression process.However, the only information actually used for anomaly detection is the product of the probabilities of the edges encountered along the path. Subsequently, with respect to a chosen threshold, an input sequence can be classified as anomalous or non-anomalous.

3.1 Dictonary building

As the authors also did, for simplicity, in this subsection we use a natural alphabet as an example to illustrate how the dictionary on which the tree is based is constructed. The existing symbols are considered to form the alphabet.

The main idea is to identify the shortest previously unseen sequences in the training data and separate them as "words".

3.2 Tree building

After constructing the dictionary, the sequences extracted from the training data are used to build a rooted tree structure. Each node in the tree represents a word, while edges correspond to symbol extensions. The root node represents the empty sequence.

We initialize the tree as to only contain the root and one child node for each symbol in the alphabet. During training, the input sequences are traversed symbol by symbol. Whenever

Algorithm 2: LZ78 Tree Construction with Counters and Probabilities

Data: Training sequences, alphabet Σ
Result: Rooted tree with counters and probabilities

Initialize root node r
Create one child of r for each symbol in Σ
foreach *training sequence* s **do**
    Set current node $v \leftarrow r$
    foreach *symbol* x *in* s **do**
        $v \leftarrow \text{child}(v, x)$
        if v *is a leaf node* **then**
            foreach *symbol* $a \in \Sigma$ **do**
                Add child node labeled a to v
            Reset $v \leftarrow r$
Assign counter 1 to each leaf node
Compute internal node counters as the sum of their children counters
foreach *node* u *in the tree* **do**
    Compute node probability: $P(u) = \dfrac{\text{count}(u)}{L}$
foreach *edge* (p, u) **do**
    Compute edge probability: $P(p \to u) = \dfrac{\text{count}(u)}{\text{count}(p)}$

a leaf node is reached, it is expanded by adding all symbols from the alphabet as its children, and the traversal is reset to the root.

A counter with value 1 is assigned to every leaf, while the counter of an internal node is defined as the sum of the counters of its children. Probabilities are assigned both to nodes and edges. The probability of a node is computed as the ratio between its counter and the total number of leaf nodes (no necessarily on the last level of the tree), while the probability of an edge is defined as the ratio between the counter of the child node and the counter of its parent.

Likelihood of new sequences during the anomaly detection stage is computed using the probabilistic tree representation.

3.3 Anomaly detection on LZ78 probabilities

The sequence is computed symbol by symbol through the tree, following the same traversal rules applied during the learning phase, this time on the test input.

For each transition in the tree, we use the probability assigned to the respective edge. The probability of the entire sequence is computed as the product of the probabilities of the edges encountered along the traversal path. Whenever a leaf node is reached, the traversal is reset to the root node, and the remaining symbols are processed in the same manner.

Based on the resulting probability value, the input sequence can be classified as normal or anomalous with respect to a predefined threshold. Sequences with low probability under the learned model are considered anomalous, while sequences with higher probability are regarded as normal.

Formally, let $x = (x_1, x_2, \ldots, x_n)$ denote an input sequence and let $\{e_1, e_2, \ldots, e_k\}$ be the sequence of edges traversed in the LZ78 tree during the parsing of x. Each edge e_i is

associated with a probability $P(e_i)$ estimated during the training phase.

The probability assigned to the input sequence is defined as

$$P(x) = \prod_{i=1}^{k} P(e_i).$$

This value represents an estimate of how likely the sequence is under theprobabilistic model learned from the training data.

Given a predefined threshold T, the anomaly detection rule is defined as

$$x = \begin{cases} \text{anomalous}, & \text{if } P(x) < T, \\ \text{normal}, & \text{otherwise.} \end{cases}$$

Sequences that yield a low probability under the learned model are therefore interpreted as deviating from the normal behavior observed during training, and are consequently flagged as anomalies.

4 Neural Compression Approach

The second approach follows the neural lossless compression framework introduced in [1]. The authors propose evaluating time series models through their ability to assign probabilities to data using lossless compression principles. The main idea is that, if a sequence is difficult to compress with a model trained on normal behavior, it is likely for it to be an anomaly.

In our implementation, we adopt this perspective by modeling the time series as a sequence of discrete symbols and estimating the probability of each symbol conditioned on its previous context using an autoregressive neural network.

4.1 Byte-level representation

In order to apply lossless compression techniques to real-valued time series, the continuous observations must first be transformed into a discrete representation. Following the approach described in the paper, each floating-point value is encoded using the IEEE 754 `float32` format, which represents a real number represented on 4 bytes.

After that, each observation is mapped to a sequence of four discrete symbols, each belonging to an alphabet of size 256. By concatenating all bytes across time steps and variables, the multivariate time series is converted into a single sequence of bytes, preserving the exact numerical information without loss.

4.2 Autoregressive probability modeling

Given the resulting byte sequence, a neural autoregressive model is trained to estimate the conditional probability of each byte given all previous bytes in the sequence. Formally, the probability assigned to a sequence $x = (x_1, x_2, \ldots, x_T)$ is factorized as

$$P(x) = \prod_{t=1}^{T} P(x_t \mid x_1, \ldots, x_{t-1}).$$

In our experiments, this conditional distribution is learned using a recurrent neural network architecture, which processes the input sequentially and produces a probability distribution over the next possible byte at each time step.

The model parameters are optimized by minimizing the negative log-likelihood of the training data, encouraging the network to assign high probability to byte sequences that corre-

Algorithm 3: Neural Byte-Level Model Training

Data: Training time series windows
Result: Trained neural compression model
Convert each window into a sequence of bytes using IEEE 754 encoding
foreach *training sequence* x **do**
    foreach *position* t *in* x **do**
        Predict x_t given $(x_1, \ldots, x_{t-1})$
        Update model parameters to maximize $P(x_t \mid x_{<t})$

spond to normal system behavior.

4.3 Training procedure

During training, fixed-length windows are extracted from the training portion of the dataset. Each window is converted into its corresponding byte sequence and used as input to the neural model. The network is trained to predict the next byte at each position, given all previous bytes.

4.4 Anomaly detection using compression probabilities

After training, the learned model is used to evaluate unseen time series windows. For each window, the probability of its corresponding byte sequence is estimated using the autoregressive factorization.

The anomaly score of a sequence is derived from its overall probability under the model. Sequences that receive low probability are considered difficult to compress and therefore indicative of abnormal behavior.

Formally, for a test sequence $x = (x_1, \ldots, x_T)$, the assigned probability is given by

$$P(x) = \prod_{t=1}^{T} P(x_t \mid x_1, \ldots, x_{t-1}).$$

Based on this value, anomaly detection is performed using a threshold-based rule:

$$x = \begin{cases} \text{anomalous}, & \text{if } P(x) < T, \\ \text{normal}, & \text{otherwise.} \end{cases}$$

Sequences that deviate significantly from the learned normal behavior are therefore detected as anomalies, as they require a larger number of bits to be represented by the model.

4.5 Interpretation

From the perspective of information theory, this approach associates abnormal behavior with low rate of compression. While normal patterns are efficiently encoded by the neural model, anomalies result in lower predicted probabilities and consequently higher encoding cost.

Here is Algorithm 1

5 Implementation and Experiments

In order to enable a fair comparison between the two compression-based anomaly detection approaches, several adaptations were required. Although both methods rely on probabilistic modeling and compression principles, they operate on different data representations and were originally proposed for different application domains. Consequently, a common experimental

Algorithm 4: LZ78 Adaptation for Time Series

Data: Time series $y_1, \ldots, y_T$
Result: Symbolic sequence
Compute temporal differences $d_t = y_t - y_{t-1}$
Normalize d_t using training statistics
Quantize values into symbols $x_t \in \Sigma$
Construct LZ78 sequences from fixed-length windows

framework was designed to evaluate both algorithms under comparable conditions.

The ETTh2 dataset was introduced in [3] and contains measurements collected from electricity transformer systems.

5.1 Common experimental setup

Both methods were applied to the same time series dataset and evaluated using identical training and testing splits. The training data was assumed to represent normal system behavior, while the testing data was used to assess the ability of each method to detect deviations from this behavior.

Since the dataset does not provide ground-truth anomaly labels, the evaluation focuses on the distribution and temporal behavior of the anomaly scores, rather than on supervised classification metrics.

In both approaches, the time series was segmented into windows of the same length.

5.2 Adaptation of the LZ78-based method

The original LZ78-based anomaly detection method was proposed for network traffic or system calls. In order to apply it to real-valued time series data, a discretization step was introduced.

First, the continuous time series was transformed using temporal differences, capturing local changes between consecutive observations. The resulting values were then standardized and mapped to a finite alphabet using clustering-based quantization. Each quantized value was treated as a symbol, allowing the construction of LZ78 sequences compatible with the original algorithm.

Formally, let y_t denote the original time series and let

$$d_t = y_t - y_{t-1}$$

be the temporal difference representation. A quantization function $q(\cdot)$ maps each d_t to a discrete symbol:

$$x_t = q(d_t), \quad x_t \in \Sigma.$$

The resulting symbolic sequence $(x_1, \ldots, x_T)$ is then processed using the LZ78 tree construction and anomaly detection procedures described in the previous section.

5.3 Adaptation of the neural compression method

The neural compression approach operates directly on discrete sequences. To preserve exact numerical, a byte-level encoding was adopted.

Each observation was encoded using the IEEE 754 `float32` representation, resulting in four bytes per value. These bytes were treated as symbols from an alphabet of size 256, producing a discrete sequence suitable for autoregressive modeling.

Unlike the original benchmark framework, which focuses on compression performance metrics, our implementation uses the learned probability estimates exclusively for anomaly detection, enabling a direct comparison with the LZ78-based method.

Formally, let z_t denote a real-valued observation. The encoding function $\phi(\cdot)$ maps it to a byte sequence:

$$(b_{t,1}, b_{t,2}, b_{t,3}, b_{t,4}) = \phi(z_t), \quad b_{t,i} \in \{0, \ldots, 255\}.$$

The concatenation of all bytes yields a discrete sequence processed by the neural model.

5.4 Unified anomaly scoring

Although the two approaches rely on different modeling strategies, both assign probabilities to sequences. To enable a direct comparison, anomaly detection is performed using a common principle: if they have a low test probability using the already trained model, they are anomalies. For both methods, the anomaly score is derived from the estimated probability $P(x)$ of a window:

$$\text{score}(x) \propto -\log P(x).$$

This unified scoring formulation allows the two methods to be compared qualitatively in terms of score distributions, temporal consistency, and sensitivity to behavioral changes, despite their different internal mechanisms.

6 Experiments

All experiments were conducted on the ETTh2 dataset, which consists of multivariate time series measurements collected from an electricity transformer system. In this work, the oil temperature (OT) channel was selected for analysis, as it represents a physically meaningful variable commonly monitored in real-world energy systems.

The dataset was divided in the order of appearance into training, validation, and testing segments using a 60%-20%-20% split. The training portion was assumed to represent normal system behavior, while the testing segment was used for anomaly detection.

Since the ETTh2 dataset does not provide ground-truth anomaly labels, the analysis focuses on the behavior and distribution of anomaly scores, rather than on supervised performance metrics such as accuracy or recall.

7 Results

This section presents the experimental evaluation of the two compression-based anomaly detection approaches investigated in this work: the LZ78 universal probability assignment method and the neural lossless compression model. As a result, the evaluation is performed in a fully unsupervised setting, focusing on the qualitative behavior of anomaly scores and their statistical properties.

7.1 Results of the LZ78-based method

The LZ78-based method assigns anomaly scores by estimating the universal probability of symbol sequences derived from the time series. Figure 2 shows the evolution of the anomaly score together with the normalized oil temperature signal.

The obtained scores exhibit smooth temporal variations, reflecting the fact that LZ78 captures long-term statistical regularities rather than short-term fluctuations, being more stable. This behavior is consistent with the theoretical foundation of universal compression, which

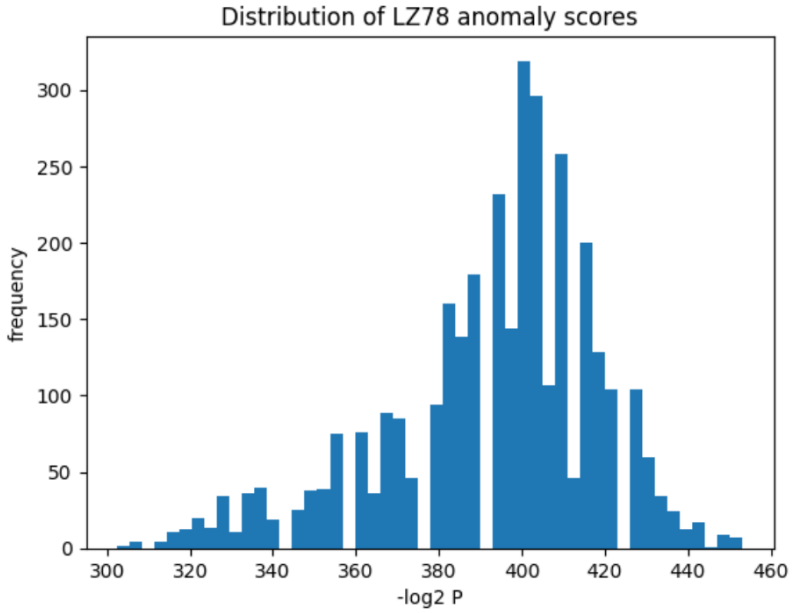

Figure 1: Distribution of LZ78 anomaly scores. The right tail corresponds to rare low-probability events.

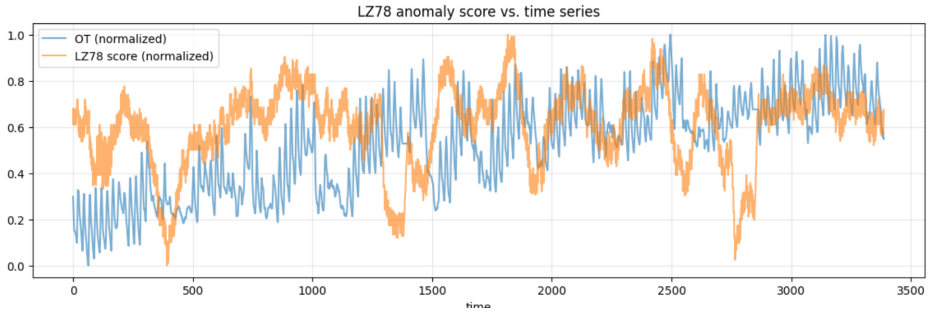

Figure 2: LZ78 anomaly score compared with the normalized oil temperature time series.

models the global structure of the data distribution.

The histogram of anomaly scores, illustrated in Figure 1, reveals a unimodal distribution with a noticeable right tail. This tail corresponds to rare patterns that are poorly represented by the learned dictionary and therefore receive lower probability values.

7.2 Results of the neural compression-based method

The neural compression approach models the byte-level representation of the time series using an autoregressive LSTM network. During training, the model exhibits stable convergence, as evidenced by the decreasing training and validation loss curves.

The compression performance achieved by the model is summarized in Table 1. The obtained bits-per-byte values indicate that the network successfully learns meaningful temporal dependencies, enabling effective probability estimation for unseen data.

Anomaly scores are computed as the negative log-likelihood of test windows. Compared to LZ78, the neural model produces sharper and more localized peaks, suggesting higher sensitivity to short-term irregularities.

Due to the lack of labeled anomalies, a percentile-based thresholding strategy was employed. Using the 95th percentile of the anomaly score distribution, approximately 5% of the

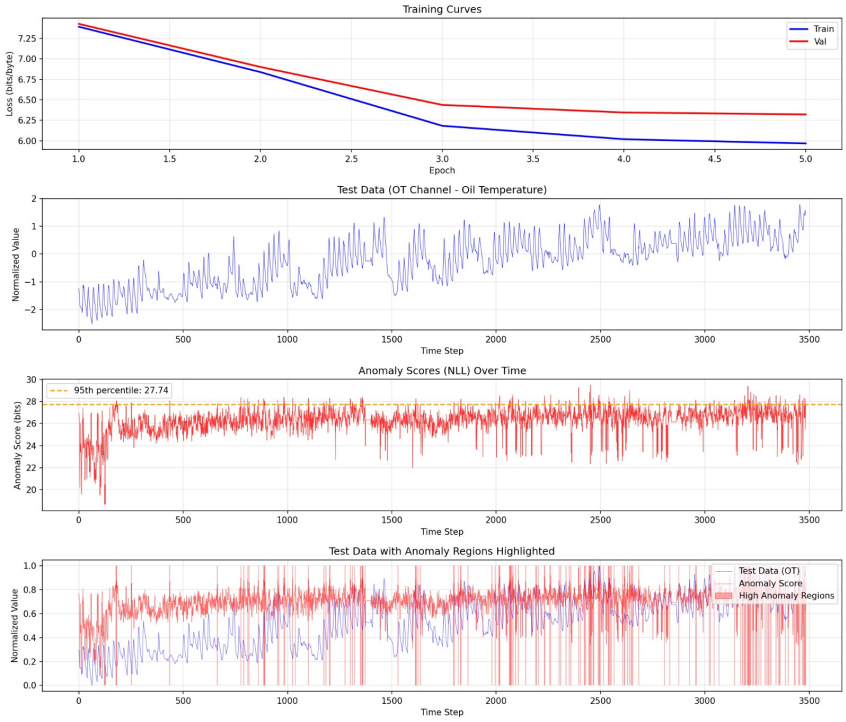

Figure 3: Neural compression results: training and validation curves, test data, and anomaly scores over time.

Table 1: Neural compression performance

Split	Loss	Bits per byte (bpb)
Validation	4.09	5.91
Test	4.25	6.13

test windows were identified as anomalies.

7.3 Statistical analysis of anomaly scores

Table 2 summarizes the main statistics of the anomaly scores produced by both approaches. While the absolute scales differ significantly because of the distinct probabilistic formulations, both methods exhibit non-trivial variance and heavy-tail behavior, which are essential for effective unsupervised detection.

7.4 Detected anomalous regions

The number of detected anomalous windows using percentile-based thresholding is reported in Table 3. Since no ground-truth labels are available, these values do not represent detection accuracy but rather characterize the sensitivity of the methods.

7.5 Comparison between the two approaches

A comparative summary of the two compression-based approaches is presented in Table 4. Although both methods rely on probability estimation through compression, they differ substantially in their modeling assumptions and temporal sensitivity.

Overall, the results demonstrate that both universal and learned compression strategies can effectively characterize normal behavior in time series data. While LZ78 provides stability

Table 2: Summary statistics of anomaly scores

Method	Mean	Std	Min	Max	Threshold
LZ78	393.23	26.11	302.51	453.14	–
Neural compression	24.48	1.64	14.24	28.83	26.90

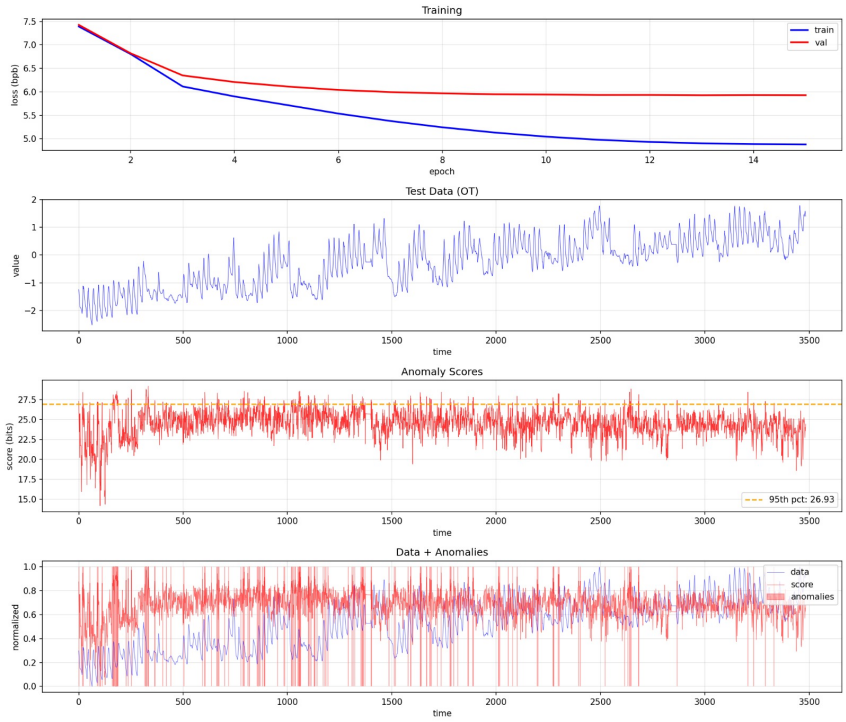

Figure 4: Overlay between normalized oil temperature and detected anomaly regions using neural compression.

and robustness to noise, the neural compression approach offers increased expressiveness and finer temporal resolution. Their complementary behavior suggests that hybrid approaches may further improve anomaly detection performance.

8 Discussion

The experimental results demonstrate that compression-based anomaly detection provides a viable unsupervised framework for time series analysis in the absence of labeled data. Both universal and learned compression strategies successfully capture deviations from normal behavior, albeit at different temporal scales.

These findings suggest that compression can serve as a general notion of normality, with anomalies emerging naturally as patterns that are difficult to encode under models learned from historical data.

9 Conclusions

In this paper, two compression-based approaches for time series anomaly detection were investigated and compared in an unsupervised setting. The first method relies on universal probability assignment using LZ78 compression, while the second approach employs neural lossless compression through autoregressive modeling.

Table 3: Detected anomalies using percentile thresholding

Method	Thresholding strategy	Detected anomalies
Neural compression	95th percentile	175 (5%)
LZ78	Distribution-based	qualitative

Table 4: Comparison between compression-based anomaly detection approaches

Property	LZ78	Neural compression
Training procedure	Tree construction	Parameter optimization
Requires labeled data	No	No
Data representation	Symbolic	Byte-level
Model type	Universal	Learned
Sensitivity	Global changes	Local anomalies
Inference complexity	Linear	Linear
Training cost	Low	Moderate

The experimental results demonstrate that both methods are capable of identifying anomalies in an unsupervised setting, although they are based on different modeling principles, both approaches assign low probability to atypical temporal patterns, supporting the interpretation of anomalies as poorly compressible events.

The LZ78-based method produces stable and smoothly varying anomaly scores, making it particularly effective for detecting long-term structural changes and regime shifts. In contrast, the neural compression approach exhibits higher sensitivity to localized irregularities, resulting in sharper anomaly peaks. This complementary behavior highlights the strengths and limitations of each method.

By bridging the gap between low-level data compression and high-level Business Intelligence, our findings provide a robust foundation for automated decision-support systems that enhance both the reliability and the economic efficiency of industrial operations.

Ultimately, the synergy between the global stability of LZ78 and the local precision of neural models suggests a robust path toward automating asset management and quality control.

10 Future Work

Several directions may be explored to further extend the directions proposed by the two papers.

First, the combination of universal and neural compression methods could be investigated, for instance by fusing their anomaly scores or using one model to guide the other. Hybrid approaches of this type may benefit from both global stability and local sensitivity.

Second, the analysis could be extended to multivariate anomaly detection by jointly modeling multiple sensor channels instead of focusing on a single variable.

Future work may include evaluation on datasets with annotated anomalies, which would make it possible to compare using standard detection metrics and facilitate a deeper analysis of detection accuracy.

References

[1] Jue Wang Cui Hui Ningming Nie1 Tiantian Liu Zongguo Wang Cao Rongqiang Peng Shi Yangang Wang Meng Wan, Benxi Tian. Lossless compression: A new benchmark for time

series model evaluation.

[2] Asaf Cohen Shachar Siboni. Universal anomaly detection: Algorithms and applications.

[3] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wenda Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of AAAI*, 2021.