Anomaly-TransformerX: Multi-Feature Fusion Ensemble with Performance-based Voting

*COMP4434 Big Data Analytics Project

Haoqian DU†
dept. of Computing
The Hong Kong Polytechnic University
Hong Kong, China
23098841d@connect.polyu.hk

Zhimeng GUO†
dept. of Applied Mathematics
The Hong Kong Polytechnic University
Hong Kong, China
22098789d@connect.polyu.hk

Huilin LIANG†
dept. of Applied Mathematics
The Hong Kong Polytechnic University
Hong Kong, China
24138797d@connect.polyu.hk

I. Grading Guide

Search problem and Dataset: Please read Section III Related Work and Section IV Dataset

Experiment Setting: Please read Section VIII Experiment Setting

Data Preprocessing: Please read Section IV Dataset B. Data Pre-processing

Paper Analysis: Please read Section III Related Work: Limitations of Anonmaly-Transformer and Section VI Anomaly Transformer

Reproduce: Please read Section VI: C. Reproduce their model

Traditional machine: Please read Section V: Traditional Machine Learning Methods

One basic Neural Network: Include in Section V: RNN and LSTM

Complicated Neural Network: Please read Section VII: Anomaly Transformer X: B.Complicated Neural Network

Sights and discussions: Please read Section IX Conclusion

***Index Terms*—Anomaly Detection, Big Data Analytics, Deep Learning, Ensemble Learning, Time Series, Transformer**

II. Introduction

Anomaly detection refers to the process of identifying outliers that significantly deviate from the majority of data [1]. This technique is widely applied in critical fields such as server monitoring [2], healthcare services [3], traffic surveillance [4], and sensor networks [5]. However, detecting anomalous activities faces multiple challenges. Firstly, anomalies are often rare and hidden within a vast amount of normal data points, making data labeling both difficult and costly. Additionally, timely detection is crucial, as any delay in identifying anomalies could jeopardize system safety and reduce operational efficiency.

For decades, anomaly detection has been a hot issue in research [6], with major contributions in data mining, machine learning, and computer vision. Machine learning techniques, particularly deep learning methods, have achieved major breakthroughs in this area. Deep learning-based anomaly detection techniques [1, 7, 8] primarily utilize neural networks to automatically learn data feature representations or anomaly scoring functions, thereby enabling more accurate anomaly identification. Researchers [9, 10, 11] have proposed numerous deep learning-based anomaly detection methods that have shown notably superior detection performance compared to traditional approaches, particularly in tackling complex detection challenges across various practical applications. In this paper, we propose our model, Anomaly-Transformer X, based on the Anomaly Transformer, addressing its limitations. Additionally, we conduct a comparative analysis of its performance against traditional machine learning and deep learning methods.

The Anomaly Transformer proposed by Xu et al. [12] is an unsupervised time series anomaly detection model based on the Transformer architecture. It utilizes an innovative Anomaly-Attention mechanism, which includes a dual-branch structure for prior and sequential associations, to capture association discrepancies. Moreover, it employs a minimax strategy to enhance the distinction between normal and abnormal patterns. However, the Anomaly Transformer has certain limitations: on one hand, it lacks a feature engineering module, making it difficult to incorporate domain knowledge; on the other hand, the dual-branch attention struggles with optimization when balancing reconstruction error and KL divergence. These challenges motivate us to propose Anomaly-Transformer X. In summary, the main contributions of this paper are as follows:

- **Architecture Simplification & Adaptation:** In Anomaly-Transformer X, we add an MLP layer to the Transformer encoder, enabling more flexible modeling of non-stationary data distributions.
- **Residual Learning via Hybrid Networks:** We implement a dual-branch residual fitting module that combines Bi-LSTM (for capturing temporal dynamics) and Tiny-ResNet (for local feature extraction) in Anomaly-Transformer X. This approach jointly captures multi-scale

anomalies that may be missed by the global Transformer.

- **Multi-Feature Fusion with Cross-Domain Attention:** We utilize a VAE encoder, a 1D CNN encoder, and an LSTM hidden layer for feature extraction, allowing the model to learn the inherent characteristics of time series data.
- **Multi-Model Ensemble with Performance-based Voting:** To further enhance the model's robustness, we adopt a weighted voting mechanism that integrates the predictions of Anomaly-Transformer X with those of other machine learning and deep learning models, resulting in a comprehensive output.

We believe that the most challenging aspect of the design of Anomaly-Transformer-X this time lies in using VAE, Bi-LSTM, and 1D-CNN to extract the features of sequential data, as well as the later feature fusion part of combining Transformer, LSTM, and Tiny ResNet.

III. Related Work

A. *Methods and Evolution in Anomaly Detection*

Anomaly detection techniques can be broadly categorized into five paradigms:

- **Clustering based:** Normal data forms dense clusters in the feature space, while anomalies are located far from these clusters or in sparse regions. Clustering-based methods mainly exploit this difference for anomaly detection. SVDD [13] and DeepSVDD [14] learn compact representations by fitting hyperspheres. THOC [15] extends this approach to hierarchical clustering, while ITAD [16] introduces a time-sensitive distance metric.
- **Forecasting-based:** Normal time series data has a predictable pattern, whereas anomalies can lead to significant deviations between predicted and actual values. Prediction-based models detect anomalies through prediction errors. The classical Vector Autoregression (VAR) model [17] allows for multivariate modeling to capture dynamic relationships but struggles with nonlinear patterns. DeepAnt [18] enhances multi-step forecasting by incorporating Convolutional Neural Networks (CNNs) to hierarchically extract local temporal patterns. RADM [19] introduces an self-attention mechanism to weight historical key points, though its computational complexity is quite high. AD-LTI [20] combines linear time-invariant models with residual analysis to improve the effectiveness of anomaly detection.
- **Density-based:** The density-based models identify low-probability regions via statistical modeling. LOF [21] and COF [22] measure local density deviations. Deep learning variants like DAGMM [23] and MPPCACD [24] model multivariate Gaussian mixtures.
- **Reconstruction-based:** Normal data can be efficiently reconstructed by models, while anomalous data, due to its deviation from normal patterns, typically results in higher reconstruction errors. LSTM-VAE [25] introduces a method that combines Variational Autoencoders (VAE) with Long Short-Term Memory networks (LSTM) to generate smoother reconstruction results. OmniAnomaly [9] extends LSTM-VAE into a stochastic recurrent neural network specifically designed for multivariate time series anomaly detection. Additionally, methods based on Generative Adversarial Networks (GANs), such as MAD-GAN [26] and FGANomaly [27], enhance the realism of generated results but also face challenges related to mode collapse.
- **Transformer-based:** In recent years, Transformers have been widely applied across various fields, including natural language processing and computer vision. They have also been introduced into the anomaly detection domain. In the Anomaly Transformer [12], an innovative Association Discrepancy architecture is proposed based on the Transformer model. Following this, TranAD [28] employs adversarial training methods, while MT-RVAE [29] combines Transformers with Variational Autoencoders (VAE). Additionally, Dual-TF [30] utilizes dual Transformer branches to model the dependencies between time series and features separately.

B. *Limitations of Anomaly-Transformer*

This paper falls under the transformer-based category. We conducted a review of the paper titled "Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy." Despite its advancements, we identify three critical issues.

*1) **Key issue one: Missuse of Ground Truth label***

In their solver.py file from 339 to 360 lines:

```
anomaly_state = False
for i in range(len(gt)):
    if gt[i] == 1 and pred[i] == 1 and not
    ↪  anomaly_state:
        anomaly_state = True
        for j in range(i, 0, -1):
            if gt[j] == 0:
                break
            else:
                if pred[j] == 0:
                    pred[j] = 1
        for j in range(i, len(gt)):
            if gt[j] == 0:
                break
            else:
                if pred[j] == 0:
                    pred[j] = 1
    elif gt[i] == 0:
        anomaly_state = False
    if anomaly_state:
        pred[i] = 1
```

They believe that anomalies will appear in clusters, so they modify the model's predictions to make all clusters identified as anomalies in the ground truth match, even when not all of the model's predicted clusters are anomalies. This approach is unacceptable and impractical in real industrial scenarios. All the baselines in their experiments adopted this detection

adjustment. **However, in our project, we did not use this relatively unreasonable and unrealistic method**.

We reproduced their experiment, attempting to comment out or not comment on the "detection adjustment." It was found that commenting out led to a significant drop in model performance, which indicates potential issues with their method. Therefore, we will propose a more robust model in the following sections.

TABLE I: Experiment Results Before and After Commenting the Code Block

Condition	Accuracy	Precision	Recall	F-score
Without Commenting	**0.9917**	**0.8899**	**0.9139**	**0.9017**
After Commenting	0.9543	0.0995	0.0125	0.0222

*2) **Key issue two: Data Pre-processing***

They exclusively utilized StandardScaler for data normalization, and no feature engineering techniques were identified (aside from the one-dimensional CNN for data embedding) to enhance the model's learning capabilities. Consequently, we will perform feature extraction and fusion prediction for the data at each time point.

*3) **Key issue three: Reconstruction Not Perfect***

Due to their min-max optimization loss design, the model may struggle to balance the reconstruction and dual-KL convergence related to prior association and series association. This could potentially undermine the robust fitting capability of transformer-based deep learning models.

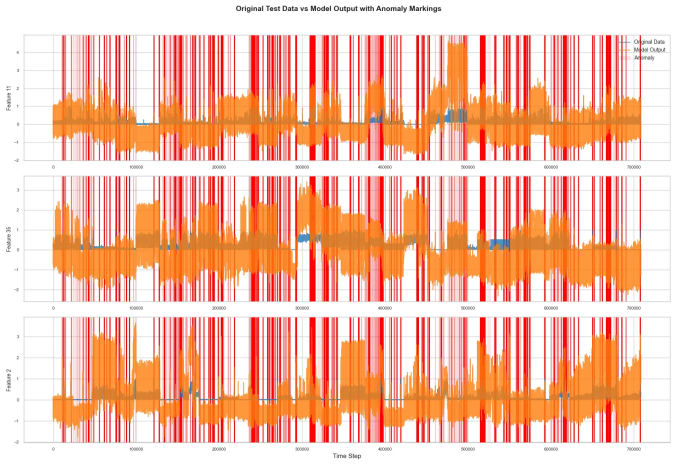


Fig. 1: Anomaly-transformer Fitting Result

As illustrated in the figure above, the predictions generated by the anomaly transformer exhibit a broader range compared to the original dataset.

We utilized their reconstruction output and employed a reconstruction-based approach to evaluate the model's performance.

We found that using only the transformer component of their anomaly transformer for data reconstruction yields higher Recall and F1 Score results compared to the methods presented in their paper. Therefore, we will also utilize the transformer for reconstruction and propose improvements.

TABLE II: Model Performance Comparison

Method	Accuracy	Precision	Recall	F1 Score
Reconstruction-Based	0.8474	0.0729	**0.2280**	**0.1105**
Their Method	**0.9543**	**0.0995**	0.0125	0.0222

As highlighted in the three key issues above, our model, anomaly-transformer X, primarily addresses these challenges through multi-modal feature extraction, the design of an complicated neural network based on the Anomaly-transformer architecture, the implementation of a reconstruction-based approach for anomaly detection, and the application of ensemble learning techniques to enhance the robustness of the final model.

IV. Dataset

We evaluate our method on the *Server Machine Dataset (SMD)*, a public benchmark for multivariate time-series anomaly detection. Multiple univariate time series from the same device (or more generally, an entity) forms a multivariate time series. SMD collects from a large Internet company, sampled at 1-minute intervals over 5 weeks. Table III and Figure 2 show the detailed description of SMD dataset.

TABLE III: Table with Automatic Width Adjustment

Dataset Name	Entities	Dims.	Training Set Size	Testing Set Size	Anomaly Ratio (%)
SMD	28	38	708,405	708,420	4.16

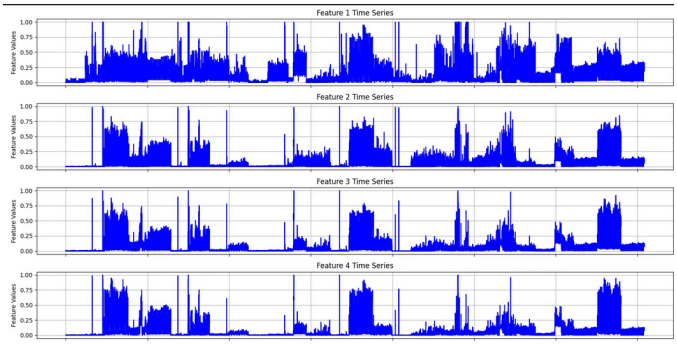


Fig. 2: SMD features

The *Server Machine Dataset (SMD)* is a widely used public dataset for anomaly detection research. Its primary characteristics are as follows:

- **Scenario Characteristics**: SMD records the operational states of multiple servers, reflecting real-world industrial environments.
- **Data Properties**:
  - It includes 28 variables (sensor metrics), such as CPU utilization and memory usage.
  - The data consists of multidimensional time series with significant temporal correlations.

- **Annotation Information**: SMD contains labeled data for both normal and anomalous samples, enabling the evaluation of models in detecting anomalies.
- **Challenges**:
  - The dataset features sparse and complex distributions of anomalies.
  - The variables are interdependent, introducing additional challenges for effective modeling.

This combination of real-world complexity and structured data makes SMD a valuable benchmark for evaluating anomaly detection methods, particularly in the context of high-dimensional, time-series data.

A. *Dataset Visualization*

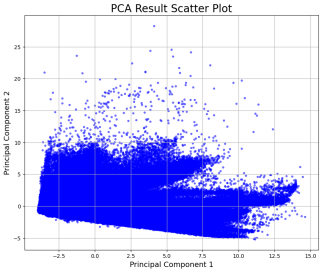


Fig. 3: 2D PCA Visualization of SMD Dataset

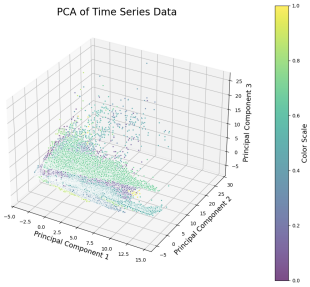


Fig. 4: 3D PCA Visualization of SMD Dataset

We performed PCA dimensionality reduction on the 38-dimensional SMD dataset to 2D and 3D for visualization. It is evident that the majority of data points are compactly clustered together, while a small number of data points are located farther away from the clusters. These outliers are highly likely to be anomalous data points. This visualization also intuitively reveals the principle behind clustering-based anomaly detection to some extent.

B. *Data Pre-processing*

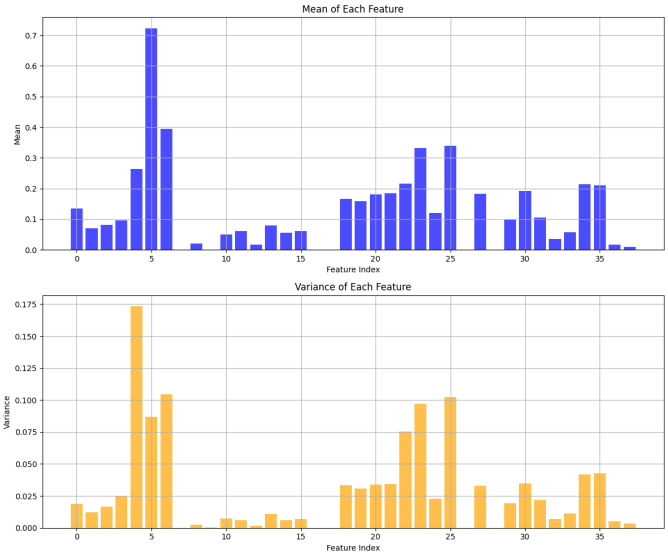


Fig. 5: Feature Statistics

We further examined the statistics of the features themselves and found significant differences in the scales of the data. Therefore, standardization is required to ensure consistency:

$$z_i = \frac{x_i - \mu}{\sigma}, \quad \text{where} \quad \mu = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Due to the unsupervised nature of the anomaly detection task and the specific characteristics of the SMD dataset, we only have unlabeled training data and labeled testing data. This means it's not suitable to perform K-fold cross-validation. The paper uses a reconstruction-based approach for anomaly detection, which can be understood as unsupervised learning or self-supervised learning, similar to VAE. Intuitively, this approach involves fitting the model to the data itself. However, the paper still splits the training data into 80% for the training set and 20% for the validation set. To ensure variable control and comparability, we use the same dataset splitting method for reconstruction-based models such as Complicated NN and LSTM, RNN. For more details, please refer to the `data-loader.py` file.

V. Traditional Machine Learning Method

A. *Density-based Methods*

1) *Local Outlier Factor*

The LOF algorithm (Local Outlier Factor) is an unsupervised outlier detection method and a representative algorithm in density-based outlier detection approaches. This algorithm calculates a Local Outlier Factor (LOF) for each point in the dataset. By determining whether the LOF is close to 1, it identifies whether a point is an outlier. If the LOF is significantly greater than 1, the point is considered an outlier; if it is close to 1, the point is regarded as normal.

As shown in the figure below:

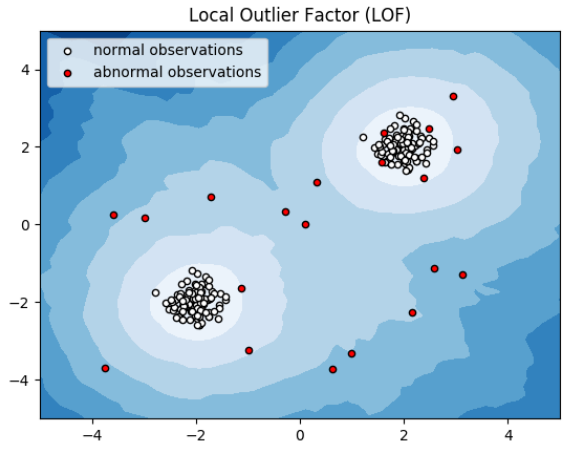


Fig. 6: LOF Concept

a) 1. k-Distance

The k-distance of a point p is defined as the distance to its k-th nearest neighbor:

$$d_k(p) = \text{distance}(p, k\text{-th nearest neighbor of } p). \quad (1)$$

b) 2. Reachability Distance

The reachability distance of a point o with respect to p is:

$$\text{reach-dist}_k(p, o) = \max\left(d_k(o), \text{distance}(p, o)\right). \quad (2)$$

c) 3. Local Reachability Density (LRD)

The local reachability density of p is the inverse of the average reachability distance of its neighbors:

$$\text{lrd}_k(p) = \frac{1}{\frac{1}{|N_k(p)|}\sum_{o \in N_k(p)} \text{reach-dist}_k(p, o)}, \quad (3)$$

where $N_k(p)$ represents the set of k-nearest neighbors of p.

d) 4. Local Outlier Factor (LOF)

The LOF score of p is the average ratio of the local reachability density of its neighbors to its own local reachability density:

$$\text{LOF}_k(p) = \frac{1}{|N_k(p)|} \sum_{o \in N_k(p)} \frac{\text{lrd}_k(o)}{\text{lrd}_k(p)}. \quad (4)$$

e) 5. Outlier Detection

- If $\text{LOF}_k(p) \approx 1$, p is considered a normal point. - If $\text{LOF}_k(p) \gg 1$, p is considered an outlier.

This method effectively identifies anomalies by comparing the density around each point with the density around its neighbors.

*2) **Isolation Forest***

The Isolation Forest algorithm is an unsupervised anomaly detection method based on the principle of isolating data points. It utilizes random partitioning to construct a forest of binary trees, where anomalies are easier to isolate due to their rarity and dispersion.

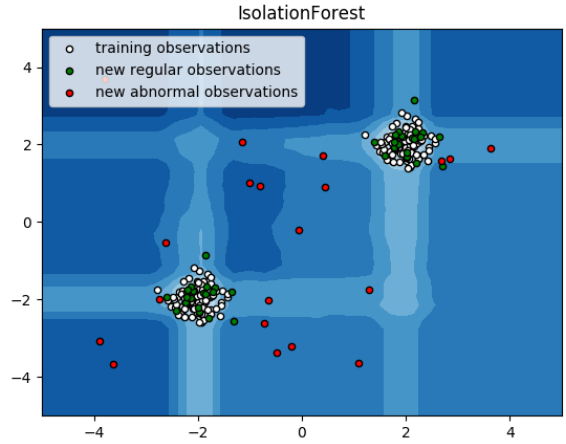


Fig. 7: Isolation Forest

a) 1. Random Partitioning

For a dataset X, the algorithm iteratively selects a random feature and splits the data along a random value within the feature's range:

$$X \xrightarrow{\text{random split}} \{X_{\text{left}}, X_{\text{right}}\}. \quad (5)$$

This process is repeated to construct a binary tree until each point is isolated or a stopping condition (e.g., tree height) is met.

b) 2. Path Length

The path length $h(x)$ of a data point x is the number of edges traversed from the root of the tree to the leaf node containing x. For a dataset with n points, the average path length of a balanced binary tree is approximated as:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}, \quad (6)$$

where $H(i)$ is the i-th harmonic number:

$$H(i) = \sum_{k=1}^{i} \frac{1}{k}. \quad (7)$$

c) 3. Anomaly Score

The anomaly score of a data point x is computed based on the path length $h(x)$ averaged over all trees in the forest:

$$s(x, n) = 2^{-\frac{\bar{h}(x)}{c(n)}}, \quad (8)$$

where $\bar{h}(x)$ is the average path length of x across all trees, and $c(n)$ normalizes the score.

d) 4. Outlier Detection

- If $s(x, n) \approx 1$, x is likely an anomaly. - If $s(x, n) \ll 1$, x is likely a normal point.

The Isolation Forest algorithm effectively isolates anomalies using fewer splits, as they are typically located in sparse regions of the feature space.

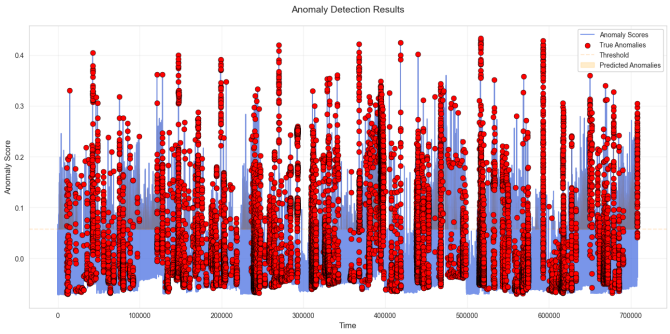


Fig. 8: Isolation Forest Prediction Visualization

B. *Clustering-based Methods*

1) *OC-SVM*

The One-Class SVM (OC-SVM) algorithm is an unsupervised anomaly detection method that identifies outliers by learning the boundary of the data distribution in a high-dimensional feature space.

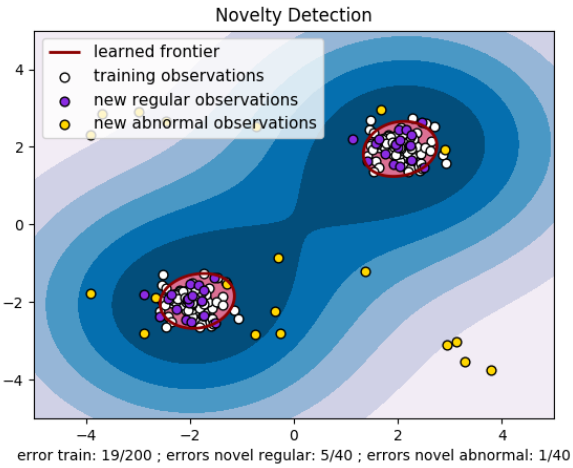


Fig. 9: OC-SVM

a) 1. Objective Function

The OC-SVM algorithm aims to separate the data points from the origin with maximum margin in a feature space transformed by a kernel function. The optimization problem is formulated as:

$$\min_{\mathbf{w},\rho,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho, \tag{9}$$

subject to:

$$(\mathbf{w}\cdot\phi(\mathbf{x}_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad i = 1,\ldots,n, \tag{10}$$

where: - $\mathbf{w}$ is the weight vector, - ρ is the offset (threshold) of the decision boundary, - ξ_i are slack variables to allow some points within the boundary, - ν is a hyperparameter controlling the trade-off between margin size and the fraction of outliers, - $\phi(\cdot)$ is the feature mapping function defined by the kernel.

b) 2. Decision Function

The decision function for a test point $\mathbf{x}$ is given by:

$$f(\mathbf{x}) = \mathbf{w}\cdot\phi(\mathbf{x}) - \rho, \tag{11}$$

which can be expressed in terms of the kernel function $K(\mathbf{x}_i,\mathbf{x})$:

$$f(\mathbf{x}) = \sum_{i=1}^{n}\alpha_i K(\mathbf{x}_i,\mathbf{x}) - \rho, \tag{12}$$

where α_i are the Lagrange multipliers obtained from solving the dual optimization problem.

c) 3. Outlier Detection

- If $f(\mathbf{x}) < 0$, the point $\mathbf{x}$ is classified as an outlier. - If $f(\mathbf{x}) \geq 0$, the point $\mathbf{x}$ is classified as normal.

d) 4. Kernel Function

The kernel function $K(\mathbf{x}_i,\mathbf{x}_j)$ defines the similarity between two points and allows the algorithm to learn a nonlinear decision boundary. Common kernel functions include:

- Linear kernel: $K(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{x}_i\cdot\mathbf{x}_j$,
- Gaussian kernel (RBF): $K(\mathbf{x}_i,\mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}\right)$,
- Polynomial kernel: $K(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{x}_i\cdot\mathbf{x}_j + c)^d$.

OC-SVM is effective for anomaly detection as it learns a flexible decision boundary that captures the majority of the data distribution while isolating outliers.

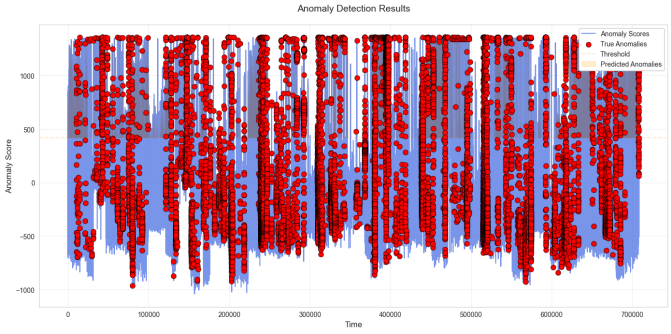


Fig. 10: OC-SVM Prediction Result Visualization

2) *Support Vector Data Description(SVDD)*

Support Vector Data Description (SVDD) is a boundary-based anomaly detection method that aims to find the smallest hypersphere in feature space that encloses the majority of the data.

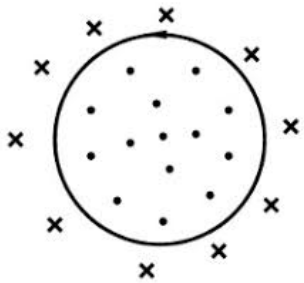


Fig. 11: SVDD

a) 1. Objective Function

SVDD seeks to minimize the radius R of the hypersphere while allowing for some data points to fall outside the boundary (controlled by slack variables). The optimization problem is formulated as:

$$\min_{R,\mathbf{c},\xi} R^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i, \tag{13}$$

subject to:

$$\|\phi(\mathbf{x}_i) - \mathbf{c}\|^2 \leq R^2 + \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n, \tag{14}$$

where: - R is the radius of the hypersphere, - $\mathbf{c}$ is the center of the hypersphere in the transformed feature space, - $\phi(\cdot)$ is the feature mapping function defined by a kernel, - ξ_i are slack variables allowing for outliers, - ν is a hyperparameter that controls the trade-off between the hypersphere's volume and the number of outliers.

b) 2. Dual Formulation

By introducing Lagrange multipliers, the problem can be reformulated into its dual form:

$$\max_{\alpha} \sum_{i=1}^{n}\alpha_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{15}$$

subject to:

$$0 \leq \alpha_i \leq \frac{1}{\nu n}, \quad \sum_{i=1}^{n}\alpha_i = 1, \tag{16}$$

where $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ is the kernel function.

c) 3. Decision Function

The decision function for a test point $\mathbf{x}$ determines whether it lies inside or outside the hypersphere:

$$f(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}) - 2\sum_{i=1}^{n}\alpha_i K(\mathbf{x}_i, \mathbf{x}) + \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j K(\mathbf{x}_i, \mathbf{x}_j). \tag{17}$$

- If $f(\mathbf{x}) \leq R^2$, the point is classified as normal. - If $f(\mathbf{x}) > R^2$, the point is classified as an outlier.

d) 4. Kernel Function

Same as OC-SVM.

e) 5. Outlier Detection

SVDD identifies points that fall outside the hypersphere (i.e., have a distance greater than R) as anomalies.

SVDD is effective for anomaly detection because it models the data boundary while tolerating outliers through the slack variables.

*3) **DeepSVDD***

Deep SVDD (Support Vector Data Description) is a deep learning-based anomaly detection method that combines the principles of traditional SVDD with the representation learning capabilities of neural networks. This approach maps data into a low-dimensional feature space using a neural network and learns the smallest hypersphere in that space which encompasses the main distribution of normal data. By minimizing the distance between the features and the center of the hypersphere, Deep SVDD can capture the intrinsic structure of the data while identifying anomalies that deviate from the hypersphere. Compared to traditional SVDD, Deep SVDD is capable of handling high-dimensional and complex data (such as images or time series) and offers stronger representation power.

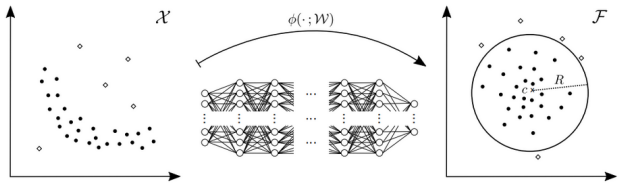


Fig. 12: Deep SVDD

Our objective is to minimize the following loss function:

$$\min_{R,\mathcal{W}} R^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\max\left\{0, \|\phi(x_i;\mathcal{W}) - c\|^2 - R^2\right\} + \frac{\lambda}{2}\sum_{\ell=1}^{L}\|\mathcal{W}^\ell\|_F^2 \tag{18}$$

In this formula, R^2 aims to minimize the radius of the hypersphere, enabling effective separation between normal and anomalous samples. $\|\phi(x_i;\mathcal{W}) - c\|^2 - R^2$ is a penalty term. When its value ¿ 0 (indicating data points fall outside the hypersphere), the overall loss increases. The term reaches its minimum (0) only when $\|\phi(x_i;\mathcal{W}) - c\|^2 - R^2 < 0$, thus encouraging the optimization to keep all normal data points within the hypersphere. The final term represents the decomposition of the L2 weight (regularization), which prevents overfitting by penalizing the weights of large networks.

C. *Reconstruction-based Methods*

1) Long Short Term Memory

We made some minor modifications to the traditional LSTM and found that using an encoder-decoder structure is more suitable for reconstruction. Therefore, the LSTM here consists of two layers: one serves as the encoder, and the other as the decoder. The two layers are stacked to perform prediction.

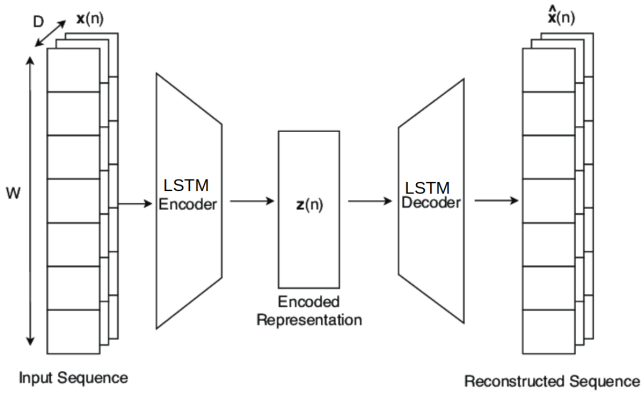


Fig. 13: LSTM-Auto Encoder

2) Recurrent neural network

Similarly, the RNN here also consists of two layers, forming an RNN-AutoEncoder structure through an encoder-decoder architecture.

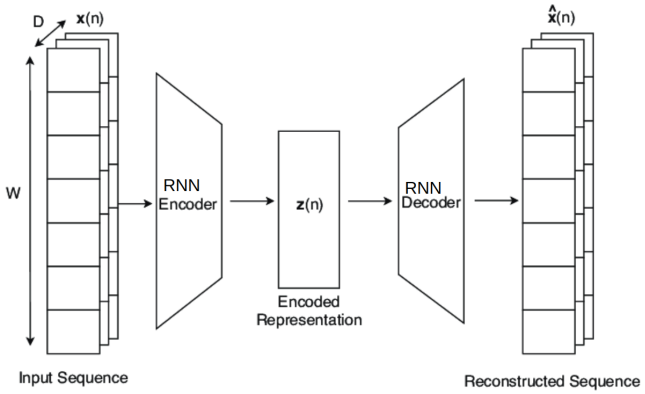


Fig. 14: RNN-Auto Encoder

D. *Forecasting-based Methods*

*1) **Autoregressive Integrated Moving Average***

ARIMA is a classical univariate time series forecasting model that combines autoregressive (AR) components, differencing (I) for trend removal, and moving average (MA) components to account for noise.

Its basic form can be represented as:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \epsilon_t$$

where: - Y_t is the current value of the time series. - ϕ_i are the autoregressive parameters. - θ_j are the moving average parameters. - ϵ_t is the white noise error term.

ARIMA is widely used for anomaly detection by analyzing residuals (the deviations between predicted and actual values), where large residuals may indicate potential anomalies. However, a key limitation of ARIMA is its inability to model interdependencies between multiple variables, necessitating separate models for each dimension (e.g., 38 independent models for the SMD dataset), which ignores cross-feature correlations and increases maintenance overhead.

*2) **Vector Autoregression***

VAR extends autoregressive modeling to multivariate time series by capturing dynamic linear relationships among variables through lagged observations.

Its basic form can be represented as:

$$\mathbf{Y}_t = \mathbf{A}_1 \mathbf{Y}_{t-1} + \mathbf{A}_2 \mathbf{Y}_{t-2} + \ldots + \mathbf{A}_p \mathbf{Y}_{t-p} + \epsilon_t \quad (19)$$

where: - $\mathbf{Y}_t$ is a vector of time series variables at time t. - $\mathbf{A}_i$ are coefficient matrices. - ϵ_t is a vector of error terms.

Anomalies are detected by measuring the Mahalanobis distance of joint residuals, which accounts for the covariance structure. While VAR excels in modeling inter-variable dependencies, it suffers from instability in high-dimensional settings due to the need to estimate large covariance matrices, leading to poor scalability for datasets with many features.

VI. Anomaly Transformer

A. *Anomaly-Attention and Loss Design*

They introduced the Anomaly-Attention mechanism featuring a two-branch architecture. In the prior-association branch, they utilize a learnable Gaussian kernel to compute the prior based on relative temporal distances. This design leverages the unimodal characteristics of the Gaussian kernel, allowing it to focus more effectively on nearby time points.

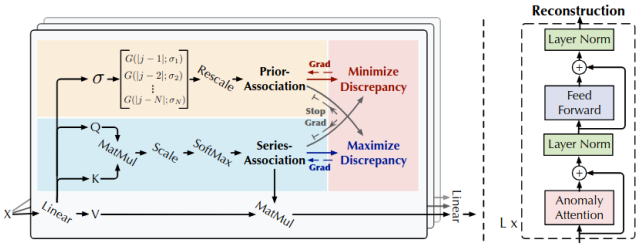


Fig. 15: Anomaly Attention

Additionally, a learnable scale parameter is employed for the Gaussian kernel, enabling the prior associations to adapt to diverse time series patterns, including varying lengths of anomaly segments. The series-association branch is designed to learn associations directly from the raw data, allowing it to identify the most effective connections in an adaptive manner. Importantly, both branches preserve the temporal dependencies of each time point, providing richer information than point-wise representations. They reflect the adjacent-concentration prior and the actual learned associations, respectively, with their differences being indicative of normal versus abnormal behavior.

The Anomaly-Attention in the l-th layer is:

Initialization:

$$Q, K, V, \sigma = X^{l-1} W^l$$

Prior-Association:

$$P_l = \text{Rescale}\left(\left[\frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{|j-i|^2}{2\sigma_i^2}\right)\right]_{i,j\in\{1,\ldots,N\}}\right)$$

Series-Association:

$$S_l = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_{\text{model}}}}\right)$$

Reconstruction:

$$\hat{Z}_l = S_l V$$

where $Q, K, V \in \mathbb{R}^{N \times d_{\text{model}}}$, $\sigma \in \mathbb{R}^{N \times 1}$ represent the query, key, value of self-attention and the learned scale respectively. $W_Q^l, W_K^l, W_V^l \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$, $W_\sigma^l \in \mathbb{R}^{d_{\text{model}} \times 1}$ represent the parameter matrices for Q, K, V, σ in the l-th layer respectively. Prior-association $P_l \in \mathbb{R}^{N \times N}$ is generated based on the learned scale $\sigma \in \mathbb{R}^{N \times 1}$ and the i-th element σ_i corresponds to the i-th time point.

Concretely, for the i-th time point, its association weight to the j-th point is calculated by the Gaussian kernel:

$$G(|j-i|;\sigma_i) = \frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{|j-i|^2}{2\sigma_i^2}\right) \text{ w.r.t. the distance } |j-$$

Furthermore, they apply Rescale$(\cdot)$ to convert the association weights into discrete distributions P_l by normalizing each row. The series-associations are represented as $S_l \in \mathbb{R}^{N\times N}$. In this context, Softmax$(\cdot)$ is used to normalize the attention map across the last dimension.

Association Discrepancy

They define the Association Discrepancy as the symmetrized KL divergence between prior and series associations, reflecting the information gain between these two distributions. To create a more informative measure, they average the association discrepancies across multiple layers, thereby integrating the associations from various levels of features as follows:

$$\text{AssDis}(P,S;X) = \frac{1}{L}\sum_{l=1}^{L}\left(KL(P_{l,i}\|S_{l,i}) + KL(S_{l,i}\|P_{l,i})\right)\Big|_{i=1}^{N}$$

where $KL(\|)$ is the KL divergence computed between two discrete distributions corresponding to every row of P_l and S_l. $\text{AssDis}(P,S;X) \in \mathbb{R}^{N\times 1}$ is the point-wise association discrepancy of X with respect to prior-association P and series-association S from multiple layers. The i-th element of results corresponds to the i-th time point of X.

From previous observations, anomalies will present smaller $\text{AssDis}(P,S;X)$ than normal time points, which makes AssDis inherently distinguishable.

B. Anomaly Criterion

Anomaly detection is an unsupervised task here. Due to the absence of ground truth, a quantitative mechanism is established for detecting anomalies. The final anomaly score leverages both the reconstruction error and the association discrepancy. The threshold is chosen by looking at the distribution of ad. Usually, it is set like the 95th percentile. If the discrepancy is higher than this threshold, it'll be labeled as anomaly.

They integrate the normalized association discrepancy into the reconstruction criterion, leveraging both temporal representation and the distinguishable association discrepancy. The final anomaly score for $X \in \mathbb{R}^{N\times d}$ is expressed as follows:

$$\text{Softmax}\left(-\text{AssDis}(P,S;X)\right) \circ \sum_{i=1}^{N}\|X_i - \hat{X}_i\|^2$$

where $\circ$ denotes element-wise multiplication. The term $\text{AnomalyScore}(X) \in \mathbb{R}^{N\times 1}$ represents the point-wise anomaly criterion for X. Typically, anomalies lead to a reduction in association discrepancy, which consequently results in a higher anomaly score. Hence, this approach enables the reconstruction error and the association discrepancy to work together, enhancing detection performance.

An intuitive explanation of their model:

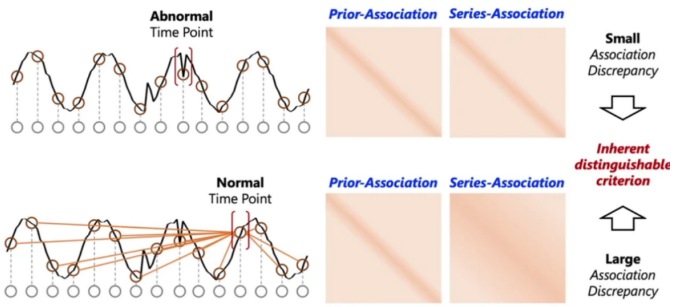


Fig. 16: Intuitive Explanition

Intuitively, if the data is normal, the distribution reconstructed by the transformer will be relatively smooth and will have a larger distance from the Gaussian distribution. In contrast, the transformer-fitted distribution for anomalous data is similar to a Gaussian distribution and will be closer in distance to it. By quantifying this difference in distance, we can predict anomalies,

C. Reproduce their model

As mentioned in the 'Limitations of Anomaly-Transformer' section of Related Work, the authors utilize detection adjustment in their code, assuming that anomalies occur in clusters. Consequently, they segment the test data and modify the model's predicted results based on the labels of each segment. While this adjustment may be a common practice in industrial applications, it is not permissible or acceptable for our project to alter predicted data using labels. Therefore, we have commented out their detection adjustment. For more information, please refer to the issue on GitHub: detection adjustment [https://github.com/thuml/Anomaly-Transformer/issues/14]."

VII. Anomaly Transformer X

A. *Preliminary model training for feature extraction*

Drawing inspiration from TimeVAE [31], we can leverage a process similar to that of autoencoders for reconstructing time series data, which inherently involves feature extraction. By continuously fitting the original data on retrained datasets, we can utilize the encoder to extract meaningful features. This approach allows us to enhance the quality of the data representation, leading to better performance in anomaly detection tasks. The extracted features will provide richer information about the underlying patterns in the time series, ultimately improving our model's ability to identify anomalies effectively.

1) *ResNet-VAE for Feature Extraction*

We initially experimented with a pure Variational Autoencoder (VAE) structure and observed that its loss function converged very slowly. To address this, we incorporated skip connection structures from ResNet to enhance the training efficiency.

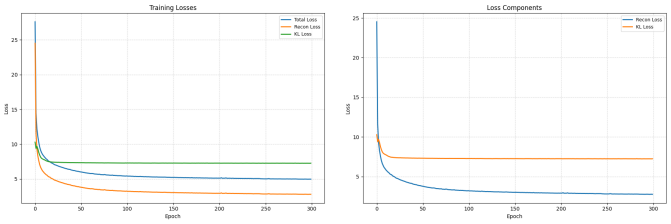


Fig. 17: VAE Loss during training. The incorporation of skip connections from ResNet significantly improved convergence speed compared to the pure VAE structure.

Our analysis revealed that the features extracted by the VAE exhibit some differences in distribution between normal and anomalous data; however, these distinctions are still not sufficiently pronounced.

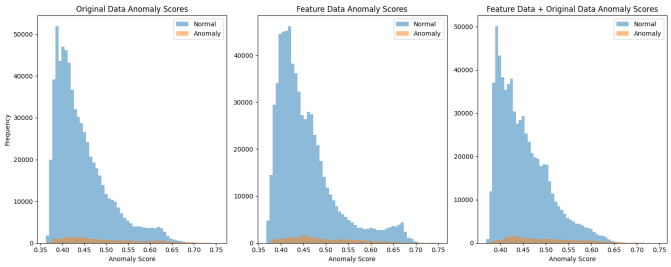


Fig. 18: VAE Anomaly Score distribution. The plot illustrates the scores assigned to both normal and anomalous data, highlighting the areas where the model struggles to differentiate effectively.

This presents our prediction results:

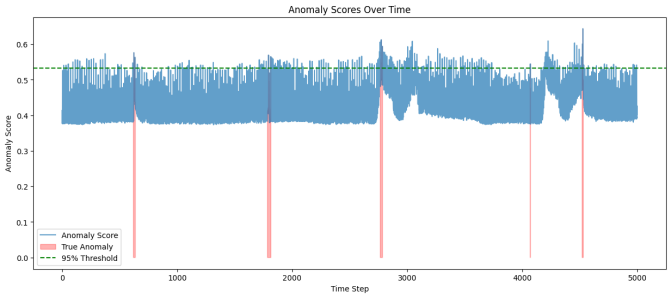


Fig. 19: Prediction Results from VAE. The graph displays the model's performance in identifying anomalies based on the predicted scores.

2) *Bi-LSTM for Feature Extraction*

We employ Bidirectional Long Short-Term Memory (Bi-LSTM) networks for both the encoder and decoder components of our model. After the convergence of the loss function, we utilize only the encoder for feature extraction, which allows us to capture relevant patterns in the input data effectively.

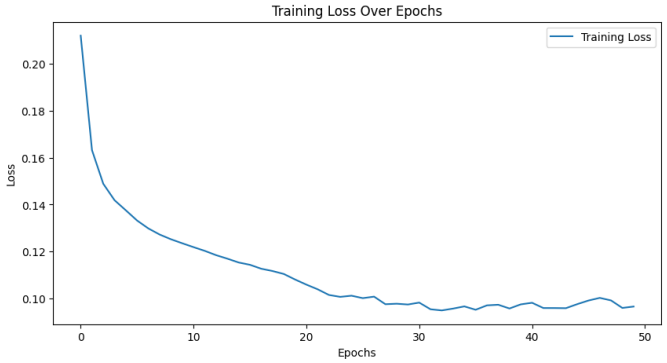


Fig. 20: Loss function behavior of the Bi-LSTM during training. The graph indicates effective convergence, demonstrating the model's learning capacity.

Using the trained Bi-LSTM, we predict anomaly scores and compare them with the true anomaly labels, as shown below. Our findings reveal that data points predicted by the Bi-LSTM with high anomaly scores are often indeed anomalies.

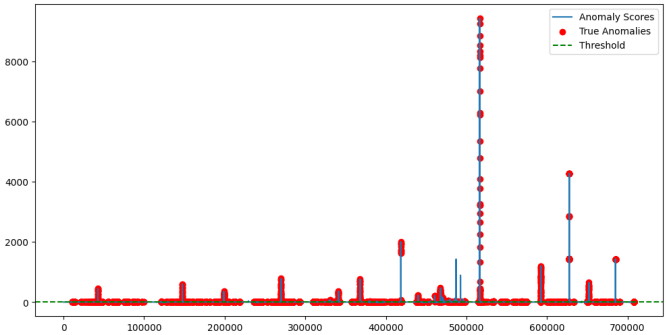
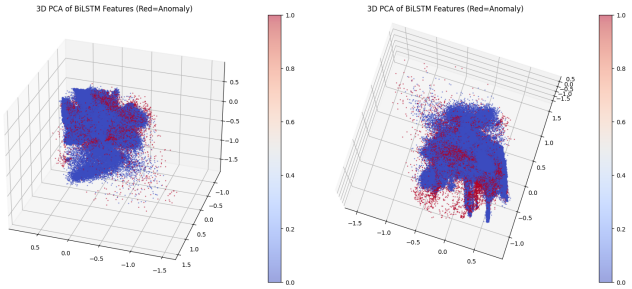


Fig. 21: Anomaly Matching Results. This figure illustrates the alignment between predicted anomaly scores and actual labels, indicating the model's effectiveness in anomaly detection.

To evaluate the quality of the extracted features, we visualize the features of both anomalous and normal data. The results demonstrate that the feature space mapped by the Bi-LSTM effectively distinguishes between anomalous and normal data.



(a) Feature space aspect 2, highlighting the distribution of normal data.

(b) Feature space aspect 3, illustrating the separation of anomalous data.

Fig. 22: Visualization of Feature Space Aspects. The plots illustrate how the Bi-LSTM separates normal and anomalous data within the learned feature space.

3) *1D-CNN (TinyResNet) for Feature Extraction*

In this section, we explore the use of a 1D Convolutional Neural Network (CNN), specifically the TinyResNet architecture, for feature extraction from time-series data. The TinyResNet model is designed to effectively capture spatial hierarchies in the data while maintaining a compact structure, making it suitable for applications with limited computational resources.

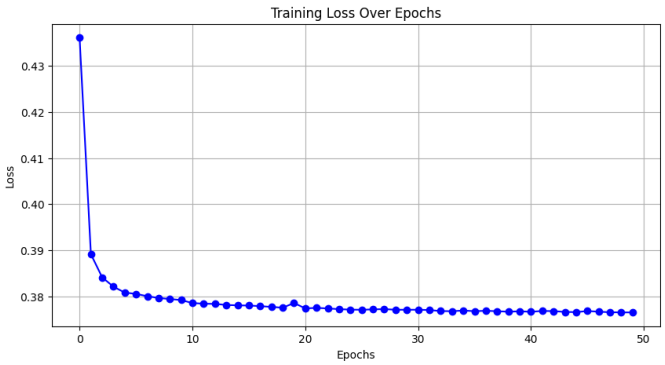


Fig. 23: Architecture of the 1D-CNN (TinyResNet) used for feature extraction. The model consists of multiple convolutional layers followed by residual connections, which help in mitigating the vanishing gradient problem and improving the overall training efficiency.

The architecture leverages residual connections to enhance the flow of gradients during backpropagation. This design choice allows the network to learn deeper representations without suffering from performance degradation typically associated with very deep networks.

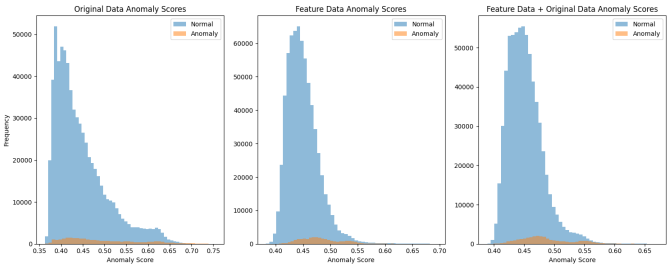


Fig. 24: Distribution of features extracted by the 1D-CNN. The plot illustrates how the model distinguishes between normal and anomalous data points based on the learned feature representations.

We visualize the distribution of the features extracted from the dataset using the TinyResNet model. The results indicate that the model is capable of differentiating between normal and anomalous data, though further tuning may be necessary to enhance this separation.

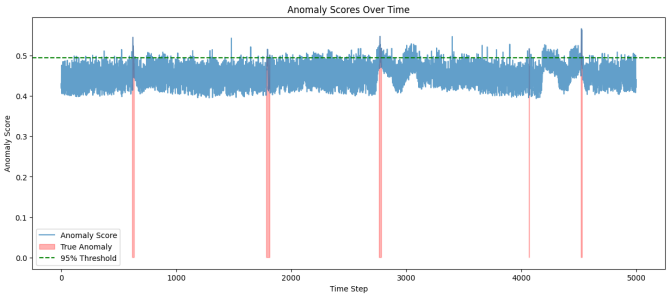


Fig. 25: Prediction results from the 1D-CNN model. The figure shows the predicted anomaly scores in relation to the true labels, demonstrating the model's performance in identifying anomalies within the dataset.

Finally, we present the prediction results, where the model's ability to correctly identify anomalies is assessed against the true labels. The outcomes suggest that the TinyResNet architecture effectively captures relevant features, leading to promising results in anomaly detection tasks.

4) *Dimension Reduction and Feature Fusion:*

The original data has a dimensionality of 38D. The Bi-LSTM reduces this to 32D, while the VAE latent attributes are also 32D. The CNN features initially have a dimensionality of 256D, which are then reduced to 32D using Principal Component Analysis (PCA). Finally, these features are concatenated horizontally to form the fusion feature.

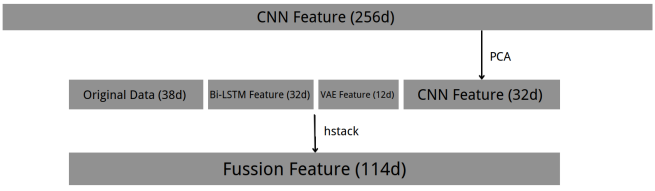


Fig. 26: Fussion Feature

B. *Complicated Neural Network(our model)*

1) *Model design motivation*

The Anomaly Transformer maximizes the Association Discrepancy based on reconstructing time series data using a Transformer. This means maximizing the dual KL divergence between the fitted prior Gaussian distribution and the series distribution fitted by the Transformer. The final Anomaly Score is determined by the following formula:

$$\text{Softmax}\left(-\text{AssDis}(P, S; X)\right) \circ \sum_{i=1}^{N} \|X_i - \hat{X}_i\|^2$$

where $\hat{X}_{i,:}$ is the reconstructed time series data, and N is the number of samples.

We calculated the reconstruction error and the association discrepancy after applying softmax, and I found that the reconstruction error is significantly larger than the association discrepancy. Therefore, simply multiplying them may lead to some issues.

Furthermore, as shown in Fig. 1, the min-max optimization strategy achieves a balance between the minimum and maximum, which may not yield excellent fitting results for the data.

To address the issue of poor data fitting, we modified the original Anomaly Transformer by retaining only the Transformer encoder and adding an MLP layer to fit the data without the prior Gaussian distribution. Inspired by Golden Noise framework[32], we introduced residual fitting for time series data. Therefore, we used Bi-LSTM and Tiny-ResNet together to fit the residual part, aiming to capture local features and temporal characteristics.

2) *Model structure design*

Loss Design:

$$X_{\text{pred}} = X_{\text{transformer}} + \alpha X_{\text{Bi-LSTM}} + \beta X_{\text{ResNet}}$$

where α and β are hyperparameters that control the weights of each model's output in the final prediction.

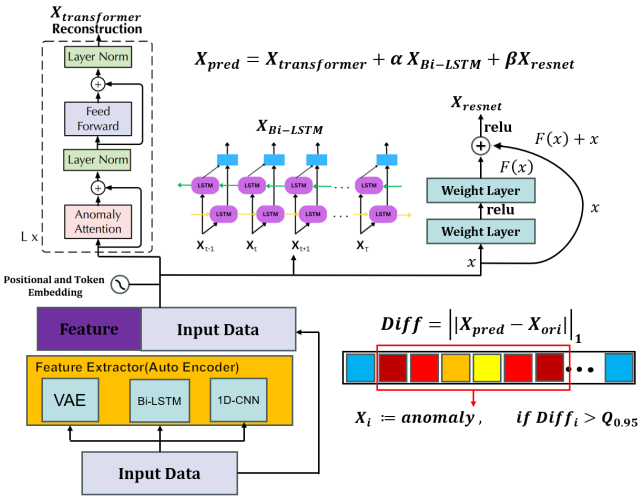


Fig. 27: Complicated Neural Network Design

3) *Model training*

Training Log:

```
(golden) haoqian@polysmart-X670E-PG-Lightning:~/anomaly/Anomaly-TransformerX$ python train.py
------------ Options -------------
alpha: 0.64
anormly_ratio: 4.0
batch_size: 1024
beta: 0.2
data_path: ./dataset/SMD
dataset: credit
input_c: 114
k: 3
lr: 0.0001
mode: train
model_save_path: checkpoints
num_epochs: 20
output_c: 38
pretrained_model: None
win_size: 100
-------------- End ----------------
====================TRAIN MODE====================
Epoch: 1 cost time: 2.4872865676879883
Epoch: 1, Steps: 7 | Train Loss: 0.3335288 Vali Loss: 0.1824007
Validation loss decreased (inf --> 0.182401).  Saving model ...
Updating learning rate to 0.0001
Epoch: 2 cost time: 1.3078052997589111
```

Fig. 28: Training Log

Testing Log:

```
(golden) haoqian@polysmart-X670E-PG-Lightning:~/anomaly/Anomaly-TransformerX$ python test.py
------------ Options -------------
alpha: 0.64
anormly_ratio: 4.0
batch_size: 1024
beta: 0.2
data_path: ./dataset/SMD
dataset: credit
input_c: 114
k: 3
lr: 0.0001
mode: test
model_save_path: checkpoints
num_epochs: 20
output_c: 38
pretrained_model: None
win_size: 100
-------------- End ----------------
====================TEST MODE====================
Precision: 0.1940
Recall: 0.2851
F1 Score: 0.2309
AUC ROC Score:0.6169
Accuracy: 0.9210
Combined Output Shape: (708400, 38)
Test Loss: 0.0830037
[0.19396432202606526, 0.2850835484309197, 0.23085808580858086, 0.6168538062175337, 0.9210446075663467]
```

Fig. 29: Model Testing

C. *Multi-model Performance Testing*

1) *Comprehensive Performance Testing*

In this section, we conduct a comprehensive performance evaluation of multiple models. We begin by defining a Softmax function to convert model scores into probability distributions. The Softmax function for a vector $\mathbf{x}$ is defined as:

$$\text{Softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Subsequently, we employ the Analytic Hierarchy Process (AHP) to calculate the weights for each model, allowing us to effectively integrate their outputs.

AHP is a structured technique for organizing and analyzing complex decisions, based on mathematics and psychology. It involves the following key steps:

1. Constructing the Decision Matrix: A pairwise comparison matrix A is created, where each entry a_{ij} represents the relative importance of criterion i over criterion j.

2. Normalizing the Matrix: Each element in the matrix is divided by the sum of its column to obtain the normalized matrix N:

$$n_{ij} = \frac{a_{ij}}{\sum_k a_{kj}}$$

3. Calculating AHP Weights: The weights w are obtained by averaging the rows of the normalized matrix:

$$w_i = \frac{1}{m} \sum_{j=1}^{m} n_{ij}$$

where m is the number of criteria.

Using the AHP weights, we assess the performance scores of each model, which are then combined to produce a final prediction.

$$\begin{bmatrix} & \text{Precision} & \text{Recall} & \text{F1} & \text{ROC-AUC} \\ \text{Precision} & 1 & \frac{1}{5} & 3 & 5 \\ \text{Recall} & 5 & 1 & 7 & 9 \\ \text{F1} & \frac{1}{3} & \frac{1}{7} & 1 & 3 \\ \text{ROC-AUC} & \frac{1}{5} & \frac{1}{3} & \frac{1}{3} & 1 \end{bmatrix}$$

Assumption: For the anomaly detection task, we consider recall to be the most important metric because it measures our ability to predict all anomalies completely. Next, we look at precision, as it reflects the accuracy of our predictions. Finally, we consider F1 Score and ROC-AUC. Based on this assumption, we derive the final decision matrix and AHP weights, and then we calculate the corresponding scores for each model based on these four performance metrics.

D. Weight Setting

The weights for the ensemble model are determined by applying the Softmax function to the calculated scores. This approach ensures that the weights reflect the relative performance of each model effectively. The Softmax transformation normalizes the scores into a probability distribution, which is subsequently used to combine the models' outputs.

E. Ensemble Learning

Ensemble learning techniques leverage the strengths of multiple models to improve overall performance. In our approach, we combine the outputs of the Complicated NN, Long Short-Term Memory-AutoEncoder (LSTM) network, Recurrent Neural Network-AutoEncoder (RNN), and Isolation Forest model. The final predictions $\mathbf{y}$ are computed as a weighted sum of the individual model outputs:

$$\mathbf{y} = \sum_{i=1}^{n} w_i \cdot \mathbf{o}_i$$

where w_i are the weights obtained from the Softmax function and $\mathbf{o}_i$ are the outputs from each model.

To generate binary classifications, we apply a thresholding rule:

$$\mathbf{y}_{\text{final}} = \begin{cases} 1 & \text{if } \mathbf{y} > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

The resulting predictions are then evaluated using various metrics, providing insights into the model's performance across different scenarios.s.

VIII. Experiments

A. Experiment Setting

1) Baselines

We compared our model with 10 baselines, including:

- **Clustering based:** OCSVM, SVDD, Deep-SVDD
- **Forecasting-based:** ARIMA, VAR
- **Density-based:** Isolation Forest, LOF
- **Reconstruction-based:** RNN, LSTM
- **Transformer-based:** Anomaly Transformer

2) Metrics

We use Precision, Recall, F1-Score and ROC-AUC four point-wise metrics to evaluate Anomaly TransformerX and other baseline models. Table IV shows the detailed information of metrics.

TABLE IV: Evaluation Metrics for Anomaly Detection

Metrics	**Formula**	**Definition**
Precision	$\frac{\text{TP}}{\text{TP} + \text{FP}}$	The proportion of correct predictions in true anomalies
Recall	$\frac{\text{TP}}{\text{TP} + \text{FN}}$	The proportion of actual anomalies that are correctly detected
F1-Score	$\frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$	Comprehensive indicator balancing both Precision and Recall
ROC-AUC	*Area under ROC curve*	Evaluates the model's overall discriminative ability across different thresholds

3) Hardware and Software

Hardware Environment: The experiments were conducted on a machine equipped with an NVIDIA GeForce RTX 4090 GPU and an AMD Ryzen 9 7900X 12-Core Processor. The system has 32GB of RAM, allowing for efficient processing of large datasets. The GPU driver version is 550.127.05, and it supports CUDA Version 12.4.

Software Environment: The implementation was carried out using Python 3.11, with the following libraries: PyTorch (version 2.6.0) for deep learning model development. NumPy for numerical computations.Pandas for data manipulation and analysis. Scikit-learn for machine learning, evaluation metrics and preprocessing.

B. Experiment Result

We conducted a series of experiments utilizing both the original dataset and the data processed through various feature extraction techniques, specifically employing 1D-Convolutional Neural Networks (1D-CNN), Bidirectional Long Short-Term Memory networks (BiLSTM), and Variational Autoencoders

(VAE) to extract meaningful features. Our approach involved not only the raw data but also the refined features to enhance the model's performance.

In our analysis, we implemented a diverse set of anomaly detection algorithms, including One-Class Support Vector Machine (OC-SVM), Isolation Forest, Local Outlier Factor, Deep Support Vector Data Description (Deep-SVDD), Autoregressive Integrated Moving Average (ARIMA), Vector Autoregression (VAR), Long Short-Term Memory Autoencoders, and RNN Autoencoders. Additionally, we integrated the model proposed in the literature, Anomaly-Transformer, which leverages transformer architecture for improved anomaly detection.

Moreover, we developed our own complex neural network architecture designed to address the unique challenges of anomaly detection. To further enhance our results, we employed an ensemble learning strategy that aggregates predictions from multiple models, including LSTM, RNN, CompNN, and Isolation Forest, utilizing a voting mechanism to determine the final prediction outcomes.

The following table presents the comprehensive data from our experiments, showcasing the performance metrics and results derived from each approach. This structured analysis aims to provide insights into the effectiveness of different models and their contributions to anomaly detection tasks.

TABLE V: Performance Metrics for SMD Datasets

Dataset	SMD			
Metric	Precision	Recall	F1	ROC-AUC
OCSVM	12.34	33.17	17.19	61.48
OCSVM*	13.92	**54.94**	22.21	70.11
IsolationForest	24.79	24.82	24.8	60.78
IsolationForest*	26.13	27.94	27.01	62.26
LOF	14.79	8.26	10.60	53.10
LOF*	18.86	7.53	10.76	53.06
SVDD*	2.89	**64.16**	5.52	35.05
Deep-SVDD*	9.57	40.67	15.49	63.60
ARIMA	9.64	21.84	13.38	61.85
ARIMA*	12.18	32.00	17.65	70.10
VAR	10.66	24.14	14.79	60.53
VAR*	13.20	34.67	19.12	71.57
LSTM	**30.61**	25.36	27.73	77.66
LSTM*	**29.83**	29.24	**29.53**	**83.57**
RNN	**29.16**	28.66	**28.91**	**79.42**
RNN*	28.26	33.75	**30.76**	**82.66**
Anomaly-transformer	9.95	1.25	2.22	50.37
Anomaly-transformer*	10.30	1.33	2.36	50.48
Complicated NN	17.85	28.84	22.06	61.54
Complicated NN*	20.89	**36.85**	26.66	65.40
Ensemble Learning	26.52	31.97	28.99	64.06

*The model name with * means using feature extraction, otherwise it means using the original data. Anomaly-transformer is the paper's model with the detection adjustment commenting out. Complicated NN is our own model.*

C. *Result Analysis*

1) *Comparison between Anomaly Transformer and comp NN*

The Anomaly Transformer achieves anomaly detection results by quantifying the distribution fitted by the transformer and the Kullback-Leibler (KL) divergence with a prior distribution, while also considering the reconstruction error. However, their model demonstrates significantly poor performance when the detection adjustment is commented out. We believe this is primarily due to the min-max training strategy employed in their model, which limits the effective utilization of the transformer for fitting data distributions.

In contrast, our approach focuses more intently on fitting the distribution of the original data. To enhance this, we incorporate Bi-LSTM for temporal feature extraction, 1D-CNN (specifically TinyResNet) for local feature extraction, and VAE for latent attribute feature extraction. These features are then fused before entering the transformer's encoder, where they undergo further feature mapping. This process generates a feature map that is richer in information.

For the classification phase, we utilize a fully connected neural network, integrating residual fitting techniques to improve performance. We employ data that has undergone transformer embedding, which is then processed through LSTM and a smaller-scale ResNet for residual fitting. The outputs from these three components are combined to form the final prediction data. This residual fitting approach maximizes the model's ability to fit the training data effectively.

As a result, our model, CompNN, significantly outperforms the Anomaly Transformer across various metrics, demonstrating improvements by several factors. This highlights the effectiveness of our method in leveraging a more robust feature extraction and fitting strategy for anomaly detection tasks.

2) *Best Model Analysis*

In the final data table, the top three values for each metric are highlighted in bold. It is important to note that while SVDD exhibits a high recall, its precision, F1 score, and ROC-AUC are all subpar, indicating that this model has low credibility due to a significant number of erroneous predictions. The best-performing models are the RNN Autoencoder and the LSTM Autoencoder. Both LSTM Autoencoders and RNN Autoencoders demonstrate exceptional performance in anomaly detection owing to their ability to handle temporal dependencies, effectively reconstruct normal data, automatically extract features, exhibit resilience to noise, dynamically adapt to changes in data distribution, and score anomalies through reconstruction errors, thereby accurately identifying data that deviates from normal patterns. Surprisingly, the OC-SVM performs exceptionally well when using extracted features, ranking just below the RNN/LSTM Autoencoders and even slightly outperforming our COMP NN model. This is attributed to the 114-dimensional input features, which, when transformed with RBF kernel to a higher dimension, align with the SVM hypothesis: points that are not linearly separable in low-dimensional space can often become linearly separable in high-dimensional space. Consequently, a very effective decision boundary can be established to separate the anomaly data, resulting in commendable performance for our OC-SVM.

D. *The goodness of feature:*

In our results table, the entries marked with an asterisk (*) indicate models that utilized extracted features, whereas those

without an asterisk used the original data. Upon comparison, we observe that each model demonstrates an improvement in performance after the incorporation of features. This finding underscores the significant role that the extracted features play in enhancing the model's ability to make accurate judgments. The enhanced performance across various metrics suggests that the features we extracted provide valuable information, facilitating better decision-making and ultimately leading to more effective anomaly detection.

IX. Conclusion

A. Insight Gained

In this project, I believe we excelled in two key areas. First, we effectively utilized VAE, Bi-LSTM, and 1D-CNN for feature extraction, which, as evidenced by our results, yielded significant improvements. Second, we innovatively applied a residual fitting approach for reconstruction, incorporating not only LSTM but also a lightweight ResNet. This combination greatly enhances our ability to fit the original data accurately.

Moreover, this valuable opportunity allowed us to conduct a detailed survey of methods in the field of anomaly detection, discussing mainstream models and their associated challenges. It also deepened our understanding of transformers and attention mechanisms. Overall, this experience has been incredibly enriching.

B. Further Work

The improvements in our COMP NN model may stem from the implementation of early feature fusion for the input data. This approach allows for the simultaneous training of the VAE, Bi-LSTM, and 1D-CNN, facilitating synchronized parameter updates. As a result, this method enhances the feature extraction process, leading to better outcomes in feature space mapping. By integrating features at an earlier stage, we can leverage the strengths of each component, ultimately improving the model's ability to learn and represent complex patterns within the data.

References

[1] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv*, 2019, preprint. [Online]. Available: https://arxiv.org/abs/1901.03407.

[2] C.-Y. Lin and S. Nadjm-Tehrani, "Protocol study and anomaly detection for server-driven traffic in scada networks," *International Journal of Critical Infrastructure Protection*, vol. 42, p. 100612, 2023, preprint. [Online]. Available: https://doi.org/10.1016/j.ijcip.2023.100612.

[3] M. Tabassum, S. Mahmood, A. Bukhari, B. Alshemaimri, A. Daud, and F. Khalique, "Anomaly-based threat detection in smart health using machine learning," *BMC Medical Informatics and Decision Making*, vol. 24, p. 347, 2024, preprint. [Online]. Available: https://doi.org/10.1186/s12911-024-02760-4.

[4] Łukasz Wawrowski, M. Michalak, A. Białas, R. Kurianowicz, M. Sikora, M. Uchroński, and A. Kajzer, "Detecting anomalies and attacks in network traffic monitoring with classification methods and xai-based explainability," *Procedia Computer Science*, vol. 192, pp. 2259–2268, 2021, preprint. [Online]. Available: https://doi.org/10.1016/j.procs.2021.08.239.

[5] R. Rizwan, F. A. Khan, H. Abbas, and S. H. Chauhdary, "Anomaly detection in wireless sensor networks using immune-based bioinspired mechanism," *International Journal of Distributed Sensor Networks*, vol. 2015, pp. 1–10, 2015, preprint. [Online]. Available: https://doi.org/10.1155/2015/684952.

[6] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, pp. 1–21, 1969, preprint. [Online]. Available: https://doi.org/10.1080/00401706.1969.10490657.

[7] S. Natha, "A systematic review of anomaly detection using machine and deep learning techniques," *Quaid-e-Awam University Research Journal of Engineering Science Technology*, vol. 20, pp. 83–94, 2022, preprint. [Online]. Available: https://doi.org/10.52584/qrj.2001.11.

[8] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection," *ACM Computing Surveys*, vol. 54, pp. 1–38, 2021, preprint. [Online]. Available: https://doi.org/10.1145/3439950.

[9] 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," *Association for Computing Machinery*, 2019, preprint. [Online]. Available: https://doi.org/10.1145/3292500.3330672.

[10] C. Bock, F.-X. Aubet, J. Gasthaus, A. Kan, M. Chen, and L. Callot, "Online time series anomaly detection with state space gaussian processes," *arXiv*, 2022, preprint. [Online]. Available: https://arxiv.org/abs/2201.06763.

[11] Z. Li, Y. Zhao, J. Han, Y. Su, R. Jiao, X. Wen, and D. Pei, "Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding," *Association for Computing Machinery*, vol. 3220–3230, 2021, preprint. [Online]. Available: https://doi.org/10.1145/3447548.3467075.

[12] J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly transformer: Time series anomaly detection with association discrepancy," *arXiv preprint arXiv:2110.02642*, 2022, version 5, last revised Jun. 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2110.02642

[13] D. M. Tax and R. P. Duin, "Support vector data description," *Machine Learning*, vol. 54, pp. 45–66, 2003, preprint. [Online]. Available: https://doi.org/10.1023/b:mach.0000008084.60811.49.

[14] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," *International Conference on Machine Learning*, pp. 4393–4402, 2018, preprint. [Online]. Available: http://proceedings.mlr.press/v80/ruff18a/ruff18a.pdf.

[15] L. Shen, Z. Li, and J. T. Kwok, "Timeseries anomaly detection using temporal hierarchical one-class network," *Neural Information Processing Systems*, vol. 33, pp. 13 016–13 026, 2020, preprint. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/97e401a02082021fd24957f852e0e475-Paper.pdf.

[16] Y. Shin, S. Lee, S. Tariq, M. S. Lee, O. Jung, D. Chung, and S. S. Woo, "Itad," *Association for Computing Machinery*, 2020, preprint. [Online]. Available: https://doi.org/10.1145/3340531.3412716.

[17] O. D. Anderson and M. Kendall, "Time-series. 2nd edn." *Journal of the Royal Statistical Society Series D (the Statistician)*, vol. 25, p. 308, 1976, preprint. [Online]. Available: https://doi.org/10.2307/2988091.

[18] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "Deepant: A deep learning approach for unsupervised anomaly detection in time series," *IEEE Access*, vol. 7, pp. 1991–2005, 2018, preprint. [Online]. Available: https://doi.org/10.1109/access.2018.2886457.

[19] N. Ding, H. Gao, H. Bu, and H. Ma, "Radm: Real-time anomaly detection in multivariate time series based on bayesian network," *IEEE Access*, vol. 35, pp. 129–134, 2018, preprint. [Online]. Available: https://doi.org/10.1109/smartiot.2018.00-13.

[20] W. Wu, L. He, W. Lin, Y. Su, Y. Cui, C. Maple, and S. Jarvis, "Developing an unsupervised real-time anomaly detection scheme for time series with multi-seasonality," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, pp. 4147–4160, 2020, preprint. [Online]. Available: https://doi.org/10.1109/tkde.2020.3035685.

[21] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof," *ACM SIGMOD Record*, vol. 29, pp. 93–104, 2000, preprint. [Online]. Available: https://doi.org/10.1145/335191.335388.

[22] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Lecture notes in computer science*, 2002, pp. 535–548, preprint. [Online]. Available: https://doi.org/10.1007/3-540-47887-6$_5$3.

[23] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International Conference on Learning Representations*, 2018, preprint. [Online]. Available: https://arxiv.org/abs/1901.04997.

[24] T. Yairi, N. Takeishi, T. Oda, Y. Nakajima, N. Nishimura, and N. Takata, "A data-driven health monitoring method for satellite housekeeping data based on probabilistic clustering and dimensionality reduction," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, pp. 1384–1401, 2017, preprint. [Online]. Available: https://doi.org/10.1109/taes.2017.2671247.

[25] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, pp. 1544–1551, 2018, preprint. [Online]. Available: https://doi.org/10.1109/lra.2018.2801475.

[26] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," *arXiv*, 2019, preprint. [Online]. Available: https://arxiv.org/abs/1901.04997.

[27] B. Du, X. Sun, J. Ye, K. Cheng, J. Wang, and L. Sun, "Gan-based anomaly detection for multivariate time series using polluted training set," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, pp. 12 208–12 219, 2021, preprint. [Online]. Available: https://doi.org/10.1109/tkde.2021.3128667.

[28] S. Tuli, G. Casale, and N. R. Jennings, "Tranad," *Proceedings of the VLDB Endowment*, vol. 15, pp. 1201–1214, 2022, preprint. [Online]. Available: https://doi.org/10.14778/3514061.3514067.

[29] X. Wang, D. Pi, X. Zhang, H. Liu, and C. Guo, "Variational transformer-based anomaly detection approach for multivariate time series," *Measurement*, vol. 191, p. 110791, 2022, preprint. [Online]. Available: https://doi.org/10.1016/j.measurement.2022.110791.

[30] Y. Nam, S. Yoon, Y. Shin, M. Bae, H. Song, J.-G. Lee, and B. S. Lee, "Breaking the time-frequency granularity discrepancy in time-series anomaly detection," *Proceedings of the ACM Web Conference 2022*, pp. 4204–4215, 2024, preprint. [Online]. Available: https://doi.org/10.1145/3589334.3645556.

[31] A. Desai, C. Freeman, Z. Wang, and I. Beaver, "Timevae: A variational auto-encoder for multivariate time series generation," *arXiv*, 2025, preprint. [Online]. Available: [Insert URL if available].

[32] Z. Zhou, S. Shao, L. Bai, Z. Xu, B. Han, and Z. Xie, "Golden noise for diffusion models: A learning framework," *arXiv*, vol. 2411.09502, 2025, submitted on 14 Nov 2024, last revised 18 Jan 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2411.09502