Multivariate Time Series Anomaly Detection based on Pre-trained Models with Dual-Attention Mechanism

Yongqian Sun*, Yang Guo*, Minghan Liang*, Xidao Wen†, Junhua Kuang*,
Shenglin Zhang*, Hongbo Li‡, Kaixu Xia‡ and Dan Pei†
*Nankai University, †Tsinghua University, ‡Tencent

***Abstract*—In major tech companies, monitoring server performance data with anomaly detection algorithms is crucial for assessing operational status. Existing models often require separate training or fine-tuning for each server due to generalization limitations, leading to increased storage, memory, and training costs. As the number of servers grows, this approach becomes impractical. To address this, we propose using pre-trained language models for time series anomaly detection, leveraging their strong generalization capabilities. Specifically, we employ two pre-trained GPT-2 models as backbones and implement a two-stage fine-tuning strategy to retain learned knowledge while adapting to specific business data characteristics. Our experiments on multiple anomaly detection datasets demonstrate that our method achieves the best average F1-Score, outperforming the leading baseline by 7%.**

***Index Terms*—Anomaly Detection, Software Reliability, Multivariate Time Series**

I. Introduction

Internet services have become increasingly integral to daily life, and the scale of the infrastructure supporting these services continues to expand [1]–[3]. Large-scale platforms deploy tens of thousands of servers, and telecom operators maintain extensive networks of base stations. Failures in individual components can degrade overall performance, negatively impacting user experience and causing significant commercial losses. Recent incidents, such as Meta's service interruption on March 5, 2024 [4], and OpenAI's outages on June 4 and June 17, 2024 [5], [6], underscore the need for robust anomaly detection systems to maintain service reliability and prevent economic losses.

To address these challenges, operation engineers configure multiple monitoring indicators to track the operational status of various components (hereafter referred to as *entities*). These indicators are collected at regular intervals, forming a time series, while data from multiple indicators form a multivariate time series (MTS). Common monitoring indicators include server metrics (CPU load, memory usage, network throughput, disk I/O), cloud service metrics (average response latency, page views, error rates), and base station metrics (wireless connection rate, traffic volume, switch success rate).

Figure 1 illustrates the multivariate time series (hereafter referred to as *MTS*) of multiple entities within a large-scale infrastructure. In real-world industrial production environments, ensuring the stable and reliable operation of tens of thousands of entities is crucial [7]. The highlighted section in the figure 1 represents an example of anomaly, where one or more indicators deviate from their normal patterns. Thus, detecting anomalies in entities essentially means identifying these deviations within the multivariate time series, pinpointing data points or segments that diverge from expected behaviors.

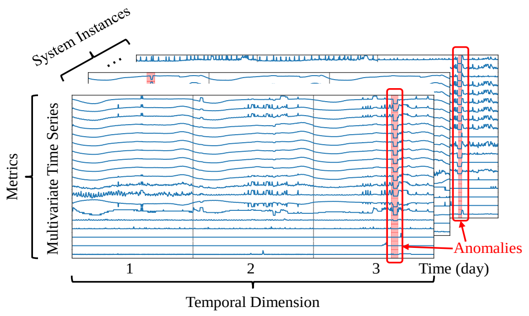

Fig. 1: The multivariate time series of system entities.

Traditional anomaly detection methods rely on operations experts setting custom thresholds for various indicators. This manual approach is impractical for the vast scale of modern internet services, requiring significant expertise and effort. Popular unsupervised techniques [8] like such as USAD [9] and OmniAnomaly [10], use structures like autoencoders (AE) [9], variational autoencoders (VAE) [10]–[13], or recurrent neural networks (RNN) [13], [14] to automate the detection. These methods, however, often necessitate separate models for each entity, leading to high storage and memory costs. Moreover, they struggle with generalization and capturing both temporal and inter-indicator relationships [13].

In contrast, models in NLP and computer vision have demonstrated strong generalization abilities [15]–[18]. The time series domain lacks such base models due to data heterogeneity, as different datasets have varying indicators and collection frequencies [19]. Despite this, research shows that pre-trained language models, like BERT, can be adapted for time series tasks through fine-tuning [20]. Leveraging this, we propose using pre-trained NLP models for time series anomaly detection. However, this approach faces two main challenges. Firstly, pre-trained language models capture relationships in one direction, while time series data requires capturing both

This research was supported by Tencent Rhino Bird Joint Research Program(No.RBFR20230705).

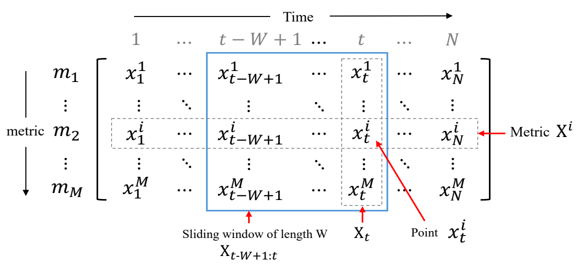

Fig. 2: Illustration of MTS data $x_t^i \in X_N^M$. Each row X^i is called a metric, while each column X_t represents the values of all metrics at a certain moment t.

temporal and inter-metric relationships. Secondly, fine-tuning must ensure the model retains pre-trained knowledge while adapting to specific domain data.

To address these challenges, we parallel two training models to solve the first challenge, and then employ a two-step fine-tuning strategy to train models to address the second challenge. Experiments show that when multiple entities share one model, our proposed anomaly detection method achieves the best results. We summarize our main contributions as follows:

- We introduce a novel approach, *DualLMAD*, which leverages pre-trained language models for anomaly detection in time series data. By addressing the distinct characteristics of MTS, such as temporal and inter-metric relationships, through a dual attention mechanism, *DualLMAD* significantly improves generalization and accuracy.
- To ensure pre-trained models retain learned knowledge while adapting to specific domain data, we propose a staged training strategy. In the first stage, only the input and output layers are updated. In the second stage, we fine-tune the add & norm layers, enhancing the model's ability to handle diverse time series datasets.
- Extensive experiments comparing with popular anomaly detection and time series algorithms across multiple datasets show that *DualLMAD* achieves superior performance. This highlights the transferability and effectiveness of pre-trained language models in time series anomaly detection. Additionally, we developed and validated an internal anomaly detection framework at a globally renowned internet company, demonstrating that *DualLMAD* meets the company's requirements for MTS anomaly detection.

II. Background

A. Problem Statement

Multivariate time series (*MTS*) consist of consecutive observations sampled at regular intervals, as illustrated in Fig. 2. Each observation $x_t^i \in X_N^M$ includes M metrics over a data length N. Metrics refer to quantifiable measures such as CPU usage and TCP active opens. In Fig. 2, the vertical axis represents different metrics, while the horizontal axis shows data collected at various timestamps. To detect anomalies at a given time point, we use a *sliding window* $X_{t-W+1:t}$, where W is the window length, combining data from the current and several previous time points.

MTS Anomaly Detection analyzes these sliding windows to determine whether the data patterns in each window deviate from historical patterns. This analysis helps us decide if the detection result should be classified as an anomaly.

B. Related Work

1) Unsupervised MTS anomaly detection: USAD [9] leverages the advantages of AE and adversarial training. A straightforward model structure and a small of parameters. OmniAnomaly [10] models explicit temporal dependence. Employs a VAE to map input observations to stochastic variables.

2) Time Series Analysis: TimesNet [21] extends the analysis of temporal variations into the 2D space by transforming the 1D time series into a set of 2D tensors based on multiple periods and then modeling the time series by 2D kernels. Time-LLM [22] re-purposes LLMs for general time series forecasting with the backbone language models kept intact. iTransformer [23], a transformer-based method utilizes the global representation of the whole series and applies attention to these metric-wise representations to capture multivariate correlations. GPT4TS [24] uses a pre-trained model for time series but neglects the metric-wise attention module.

3) Transfer Learning: OmniTransfer [25], a model-agnostic framework that combines weighted hierarchical agglomerative clustering with an adaptive transfer learning strategy, making MTS anomaly detection models efficient and effective. However, a model for each entity still needs to be saved.

C. Motivation

Deploying anomaly detection systems for tens of thousands of machines creates significant storage and memory overhead due to the need for separate models for each entity. Managing these models involves maintaining a complex table of entities and their corresponding models. During detection, loading models increases disk I/O, while keeping all models in memory incurs substantial overhead, limiting the number of entities a single node can handle.

Therefore, for MTS anomaly detection, we aim to develop a more generalized model that can be trained on historical data from multiple entities. Once trained, this model should perform accurate anomaly detection across multiple entities.

III. Approach

A. Model Structure

DualLMAD employs two pre-trained GPT-2 [18] models to simultaneously capture temporal dependencies within the time series and inter-metric dependencies. Each model is dedicated to one dimension of the data, and their outputs are merged for decoding. This dual approach allows us to comprehensively understand both the time-based and metric-based relationships within the data. We use GPT-2 in this work because the heterogeneity of time series data presents significant challenges in organizing a large-scale training set for a base model, especially when considering pretrained time

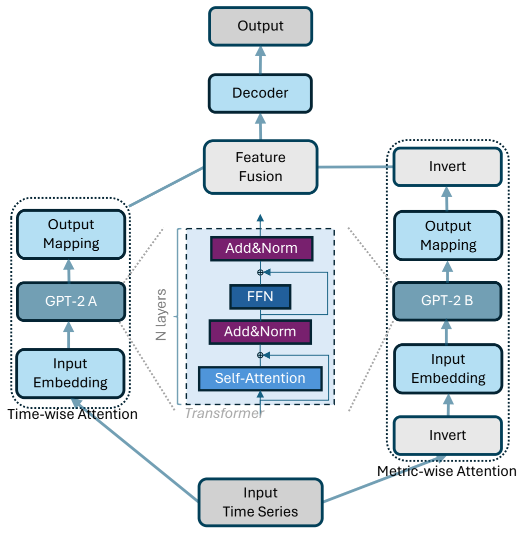

Fig. 3: Overview of *DualLMAD*.

series foundation models like MOMENT [26], Moirai [27], TimesFM [28]. These models, while advanced, are often not tailored for anomaly detection [26] and tend to ignore critical inter-metric relationships. Should more advanced pre-trained models for anomaly detection become available, *DualLMAD* is designed for easy replacement of GPT-2 to further enhance performance. The structure of *DualLMAD* is depicted in Fig. 3.

Input embedding To adapt time series data for a pre-trained model, we introduce an input embedding layer that maps the dimensionality of the time series to a dimension acceptable to the pre-trained model. A fully connected layer is introduced for this purpose. Additionally, we add position embeddings for tokens to capture temporal attention since temporal sequence differences are meaningful. However, we do not add position embeddings for inter-metric attention, as the order of metrics does not impact the final prediction.

Time-wise self attention We use the transformer layers from a pre-trained model as the backbone. Suppose the dimension of a time window is $M \times W$, where M is the number of metrics and W is the window size. After passing the data at each time point through the input embedding layer, we obtain a result of $W \times D$, where D is the hidden layer dimension of the pre-trained model (e.g., 768 in the case of GPT-2). This $W \times D$ output serves as the input to GPT-2 to capture temporal dependencies. We only fine-tune the add & norm layers of the transformers [24].

Metric-wise self attention To capture inter-metric dependencies, we transpose the $M \times W$ input to $W \times M$, treating each metric as a token, following [23], [29]. After applying input embedding, we obtain $M \times D$, which is fed into the pre-trained model to extract relationships between metrics. Unlike temporal attention, the order of metrics is arbitrary and does not affect prediction results. Thus, we do not apply position embeddings to different metrics before feeding them into the transformer model.

Feature fusion Finally, we concatenate the information extracted from the two GPT-2 models to form the final encoding, which is then used for decoding. A fully connected layer serves as the decoding layer.

TABLE I: Dataset Information.

Dataset	# entities	# metrics	time points Train	time points Test
SMD	28	38	708405	708420
SMAP	55	25	135183	427617
MSL	27	55	58317	737729
Data1	200	19	134400	672*200
Data2	200	25	288000	576*200

B. Offline Training

For machines deployed with the same type of business, we collect data over a period of 7 to 14 days. Missing values are replaced using linear interpolation, and the dataset is standardized using a standard scaler. To enable the use of a single model for detecting anomalies across multiple entities, we combine data from these machines into a single training set. During the training phase, we train *DualLMAD* to learn the historical data patterns of multiple entities, aiming to minimize the reconstruction error. To achieve optimal training results, we employ a two-step fine-tuning strategy. In the first stage, we freeze all parameters of the two GPT-2 models and train only the input-output mapping and decoder parameters. This method prevents the pre-trained model from forgetting its learned knowledge while accommodating updates in the input-output mapping and decoder parameters. In the second stage, we fine-tune the add&norm layers of the pre-trained model to better adapt to specific business scenarios.

C. Online Detection

The model reconstructs the input time series and computes the reconstruction error, which serves as the anomaly score. To dynamically determine the anomaly threshold, we apply the SPOT algorithm [30], which is effective for its ability to adapt to varying data distributions. If the anomaly score exceeds this threshold, the instance is classified as an anomaly. To further enhance efficiency, we batch data windows from different entities leveraging the model's capacity to handle multiple inputs simultaneously [29].

IV. Evaluation

In this section, we answer the following research questions:

RQ1 Performance Comparison: How does *DualLMAD* compare to existing anomaly detection methods and state-of-the-art time series algorithms?

RQ2 Design Effectiveness: How effective are the individual design components of *DualLMAD*?

RQ3 Production Performance: Can *DualLMAD* effectively perform anomaly detection in a real-world environment?

TABLE II: Best F1 Scores for *DualLMAD* and Baselines.

Method	SMD	SMAP	MSL	Data1	Data2
GPT4TS [24]	0.8481	0.6887	0.8415	0.648	0.7511
TimesNet [21]	0.8457	0.6972	0.8184	0.7004	0.8146
iTransformer [23]	0.7119	0.6935	0.7254	0.6081	0.7793
USAD [9]	0.7892	0.6994	**0.8849**	0.2611	0.3653
OmniAnomaly [10]	0.6233	0.7036	0.8257	0.1767	0.3768
DualLMAD	**0.8661**	**0.7246**	0.8739	**0.8308**	**0.9180**

A. Experimental Setup

Dataset We selected three publicly available datasets—SMD [10], SMAP [31], and MSL [31]—and two private datasets, Data1 and Data2, to validate the performance of *DualLMAD*. The statistics of these datasets are shown in Table I. SMD is a server machine dataset. SMAP and MSL are datasets publicly released by NASA. SMAP contains soil samples and telemetry data while MSL dataset includes sensor data from the Mars rover. Data1 is collected from a global content service provider, while Data2 consists of metric data collected from different base stations in a specific region by a network supplier.

Baselines For comparison, we selected two unsupervised anomaly detection algorithms: USAD [9] and OmniAnomaly [10]. Additionally, we included three popular time series algorithms: TimesNet [21], iTransformer [23], and GPT4TS [24]. These baselines provide a comprehensive benchmark for evaluating the performance of *DualLMAD*.

Evaluation metrics Anomaly detection is a binary classification task where both precision and recall are crucial. High precision avoids false alarms, while high recall ensures anomalies are detected. Therefore, we use the F1 score, the harmonic mean of precision and recall, as our performance metric. In our experiments, we use point-adjustment [32], which considers an entire anomaly segment correctly predicted if at least one point within it is correctly identified. This approach is practical in production, as a single alert during an anomaly is enough to notify maintenance staff.

Implementation *DualLMAD* is implemented in Python 3.8 using the PyTorch framework. The source code is publicly available [33]. All experiments were conducted on a Linux server equipped with 64 cores, two NVIDIA V100 32GB GPUs, and 192 GB of RAM.

B. Overall Performance (RQ1)

Table II presents the F1 scores of *DualLMAD* in comparison to baseline methods. For public datasets, all entities were combined into a single dataset for training and testing, and the overall F1 score was calculated from the merged test sets. For private datasets with numerous entities, we combined all training entities into one dataset and trained the model accordingly. Each entity was then evaluated separately, calculating TP, TN, FN, and FP for each before computing the total F1 score. To ensure a fair comparison, various anomaly thresholds were tested for each model, and the highest F1 score

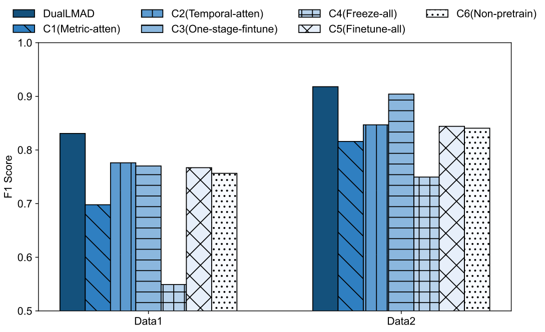

Fig. 4: Performance of *DualLMAD* and Its Variants.

was reported. Merging entities for training and testing aimed to enhance the model's generalization capability, ensuring robust performance across diverse datasets.

Overall, *DualLMAD* achieved the best results. While popular algorithms performed adequately on smaller datasets like SMD, SMAP, and MSL, their performance declined on larger datasets such as data1 and data2, each with 200 entities. This decline is due to these algorithms being designed for individual entity training, leading to poor generalization and an inability to learn patterns from multiple entities. Other time series algorithms like TimesNet and iTransformer also showed inferior performance compared to *DualLMAD* due to not leveraging extensive pre-trained knowledge. Additionally, iTransformer lacks an effective attention mechanism for capturing temporal information. GPT4TS, although using a pre-trained language model, only extracts information along the time dimension and does not capture inter-metric relationships or employ a two-step fine-tuning strategy, contributing to its lower performance.

It is worth noting that despite the use of a dual-attention mechanism, the training and detection efficiency of *DualLMAD* remain within the same magnitude as those of other baseline algorithms. Details are omitted to conserve space.

C. Ablation Study (RQ2)

To demonstrate the effectiveness of different components in *DualLMAD*, we designed the following variants (Fig. 4):

- **C1** uses a single GPT-2 model to capture inter-metric relationships. This tests the importance of capturing relationships between different metrics.
- **C2** uses a single GPT-2 model to capture temporal relationships. This evaluates the significance of modeling temporal dependencies.
- **C3** freezes the self-attention and FFN layers, performing a single-stage fine-tuning. This variant examines the impact of the two-stage fine-tuning strategy.
- **C4** freezes all parameters in the pre-trained models. This tests the baseline performance without any fine-tuning.
- **C5** fine-tunes all parameters in the pre-trained models during the second training stage. This tests the effect of fine-tuning all parameters in the second stage.
- **C6** randomly initializes the parameters in the pre-trained models. This variant serves as a control to evaluate the benefit of pre-training.

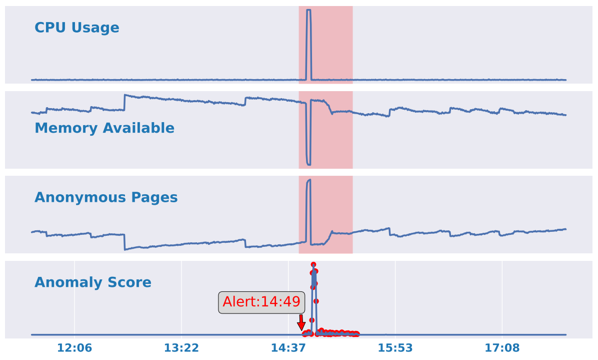

Fig. 5: NoSQL Storage Business Cyber Exercise. *DualLMAD* detected the anomaly by identifying significant deviations in memory and CPU metrics.

Dual-attention mechanism C1 and C2 each used a single attention mechanisms, and their performance was not as good as *DualLMAD*. Dual-attention mechanisms provides a more comprehensive understanding of the temporal dynamics and inter-metric relationships in the time series data. This leads to better reconstruction and anomaly detection.

Two-stage fine-tuning strategy C3, C4, and C5 employed different fine-tuning strategies, but none matched the performance of the two-stage fine-tuning strategy used by *DualLMAD*. In the first stage, C3 simultaneously fine-tuned the output mapping layer and the parameters of the add&norm layers. However, because the output mapping layer was initially randomly initialized, this could lead to suboptimal optimization of the add&norm layers. C4, by freezing all pre-trained model parameters, could not adapt to specific data, resulting in reduced flexibility and performance. In contrast, C5, fine-tuning all parameters, risked overfitting to new data and losing a significant amount of knowledge learned during pre-training.

Fine-tuning based on pre-trained models The results of C6 underscore the necessity of fine-tuning based on a pre-trained model. They demonstrate that the knowledge learned from language data by the pre-trained model is transferable to time series data, highlighting the effectiveness of leveraging pre-trained models for anomaly detection in time series.

D. Case Study (RQ3)

To verify the effectiveness of *DualLMAD* in a production environment, we conducted two case studies:

a) ***NoSQL-Storage Business Cyber Exercise:*** Monthly drills involve injecting faults into production machines to test service robustness. During the drill on January 26th, faults were injected into a designated machine. Fig. 5 displays system metrics used by *DualLMAD* for detection and the real-time anomaly scores reported by the algorithm. Significant anomalies were observed in memory (remaining memory) and CPU metrics before and after the fault injection. The anomaly scores generated by *DualLMAD* began to rise significantly from 14:49, triggering an alert. This demonstrates that *DualLMAD* successfully detected the anomaly, validating its effectiveness in a real-world scenario.

b) ***Cache Penetration:*** As the company extensively utilizes caching technologies across its operations, improper

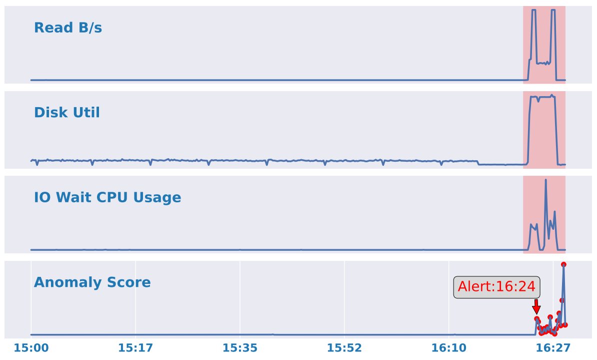

Fig. 6: Cache Penetration. After improper memory recycling strategies led to cache penetration risks, *DualLMAD* enabled operations personnel to quickly identify and address the issue.

memory recycling strategies can sometimes lead to cache penetration risks. We conducted validation experiments for this scenario. Fig. 6 shows relevant metrics and anomaly scores reported by *DualLMAD*. Upon detecting anomalies, we recommend specific metrics based on their reconstruction error. During cache penetration incidents, memory pages used by the business cannot be found in memory and must be read from disk, causing significant increases in metrics such as read_bps (read bytes per second) and disk_util (disk utilization). Our accurate recommendation of metrics such as read_bps and disk_util allows operations personnel to quickly identify cache penetration issues, distinguishing them from network or scheduling failures.

These studies demonstrate that *DualLMAD* can effectively detect anomalies in a production environment, by validating its practical applicability and reliability. However, potential limitations can include the dependency on the quality of collected metrics and the adaptability of *DualLMAD* to new, unseen scenarios or drastic changes in the operational environment.

V. Conclusion

Traditional anomaly detection frameworks typically use small models, requiring a separate model for each entity. To enhance model generalization and enable a single model to detect anomalies across multiple entities, we leveraged pre-trained models from the language domain as the backbone. These models serve as robust checkpoints, which we fine-tuned to adapt to specific application scenarios. Recognizing the unique characteristics of time series data, involving both temporal and inter-metric relationships, we employed dual pre-trained models in parallel to effectively captures comprehensive information. Our extensive experiments on three public and two private datasets demonstrated superior performance compared to existing methods. Furthermore, case studies in real-world production environments validated its practicality and effectiveness. These findings highlight the potential for using pre-trained language models in time series anomaly detection and open avenues for future research, such as exploring other pre-trained models and applying this method to various types of time series data.

References

[1] Y. Su, Y. Zhao, W. Xia, R. Liu, J. Bu, J. Zhu, Y. Cao, H. Li, C. Niu, Y. Zhang, Z. Wang, and D. Pei, "Coflux: Robustly correlating kpis by fluctuations for service troubleshooting," 2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS), pp. 1–10, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:189926947

[2] M. Ma, J. Xu, Y. Wang, P. Chen, Z. Zhang, and P. Wang, "Automap: Diagnose your microservice-based web applications automatically," in Proceedings of The Web Conference 2020, 2020, pp. 246–258.

[3] A. Borghesi, M. Molan, M. Milano, and A. Bartolini, "Anomaly detection and anticipation in high performance computing systems," IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 4, pp. 739–750, 2021.

[4] thousandeyes, "Meta outage analysis: March 5, 2024," 2024. [Online]. Available: https://www.thousandeyes.com/blog/meta-outage-analysis-march-5-2024

[5] OpenAI, "06-04-2024-chatgpt-widespread-outage," 2024. [Online]. Available: https://community.openai.com/t/06-04-2024-chatgpt-widespread-outage/801592r

[6] TechRadar, "Chatgpt was down again – here's what you need to know about the outage," 2024. [Online]. Available: https://www.techradar.com/news/live/chatgpt-is-down-heres-what-we-know-about-the-outage-so-far

[7] Y.-L. Lee, D.-C. Juan, X.-A. Tseng, Y.-T. Chen, and S.-C. Chang, "Dc-prophet: Predicting catastrophic machine failures in d ata c enters," in Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part III 10. Springer, 2017, pp. 64–76.

[8] Z. Chen, Z. Peng, X. Zou, and H. Sun, "Deep learning based anomaly detection for muti-dimensional time series: A survey," in China Cyber Security Annual Conference. Springer Nature Singapore Singapore, 2021, pp. 71–92.

[9] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "USAD: unsupervised anomaly detection on multivariate time series," in KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020, R. Gupta, Y. Liu, J. Tang, and B. A. Prakash, Eds. ACM, 2020, pp. 3395–3404. [Online]. Available: https://doi.org/10.1145/3394486.3403392

[10] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019, A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, Eds. ACM, 2019, pp. 2828–2837. [Online]. Available: https://doi.org/10.1145/3292500.3330672

[11] L. Dai, T. Lin, C. Liu, B. Jiang, Y. Liu, Z. Xu, and Z.-L. Zhang, "Sdfvae: Static and dynamic factorized vae for anomaly detection of multivariate cdn kpis," in Proceedings of the Web Conference 2021, 2021, pp. 3076–3086.

[12] Y. Su, Y. Zhao, M. Sun, S. Zhang, X. Wen, Y. Zhang, X. Liu, X. Liu, J. Tang, W. Wu, and D. Pei, "Detecting outlier machine instances through gaussian mixture variational autoencoder with one dimensional CNN," IEEE Trans. Computers, vol. 71, no. 4, pp. 892–905, 2022. [Online]. Available: https://doi.org/10.1109/TC.2021.3065073

[13] 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," in KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021, F. Zhu, B. C. Ooi, and C. Miao, Eds. ACM, 2021, pp. 3220–3230. [Online]. Available: https://doi.org/10.1145/3447548.3467075

[14] N. Zhao, J. Chen, Z. Yu, H. Wang, J. Li, B. Qiu, H. Xu, W. Zhang, K. Sui, and D. Pei, "Identifying bad software changes via multimodal anomaly detection for online service systems," in Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2021, pp. 527–539.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, vol. 30, 2017.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[17] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly et al., "An image is worth 16x16 words: Transformers for image recognition at scale," arXiv preprint arXiv:2010.11929, 2020.

[18] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever et al., "Language models are unsupervised multitask learners," OpenAI blog, vol. 1, no. 8, p. 9, 2019.

[19] R. Godahewa, C. Bergmeir, G. I. Webb, R. J. Hyndman, and P. Montero-Manso, "Monash time series forecasting archive," arXiv preprint arXiv:2105.06643, 2021.

[20] W. Dang, B. Zhou, L. Wei, W. Zhang, Z. Yang, and S. Hu, "Ts-bert: Time series anomaly detection via pre-training model bert," in Computational Science–ICCS 2021: 21st International Conference, Krakow, Poland, June 16–18, 2021, Proceedings, Part II 21. Springer, 2021, pp. 209–223.

[21] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "Timesnet: Temporal 2d-variation modeling for general time series analysis," in The eleventh international conference on learning representations, 2022.

[22] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan et al., "Time-llm: Time series forecasting by reprogramming large language models," arXiv preprint arXiv:2310.01728, 2023.

[23] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "itransformer: Inverted transformers are effective for time series forecasting," arXiv preprint arXiv:2310.06625, 2023.

[24] T. Zhou, P. Niu, L. Sun, R. Jin et al., "One fits all: Power general time series analysis by pretrained lm," Advances in neural information processing systems, vol. 36, pp. 43 322–43 355, 2023.

[25] Y. Sun, M. Liang, Z. Che, D. Li, T. Zheng, S. Zhang, P. Zhu, Y. Zhang, and D. Pei, "Efficient multivariate time series anomaly detection through transfer learning for large-scale web services," in 2023 IEEE International Conference on Web Services (ICWS). IEEE, 2023, pp. 145–151.

[26] M. Goswami, K. Szafer, A. Choudhry, Y. Cai, S. Li, and A. Dubrawski, "Moment: A family of open time-series foundation models," arXiv preprint arXiv:2402.03885, 2024.

[27] G. Woo, C. Liu, A. Kumar, C. Xiong, S. Savarese, and D. Sahoo, "Unified training of universal time series forecasting transformers," arXiv preprint arXiv:2402.02592, 2024.

[28] A. Das, W. Kong, R. Sen, and Y. Zhou, "A decoder-only foundation model for time-series forecasting," arXiv preprint arXiv:2310.10688, 2023.

[29] Z. He, P. Chen, and T. Huang, "Share or not share? towards the practicability of deep models for unsupervised anomaly detection in modern online systems," in 2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2022, pp. 25–35.

[30] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, "Anomaly detection in streams with extreme value theory," in Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 1067–1075.

[31] "Nasa datasets from omnianomaly github," 2024. [Online]. Available: https://github.com/NetManAIOps/OmniAnomaly

[32] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018, P. Champin, F. Gandon, M. Lalmas, and P. G. Ipeirotis, Eds. ACM, 2018, pp. 187–196. [Online]. Available: https://doi.org/10.1145/3178876.3185996

[33] DualLMAD, "Open source repository of duallmad," 2024. [Online]. Available: https://github.com/guo602/DualLMAD