

PatchTrAD: A Patch-Based Transformer focusing on Patch-Wise Reconstruction Error for Time Series Anomaly Detection

Samy-Melwan Vilhes
 INSA Rouen, Univ Rouen
 Normandie Univ, LITIS UR 4108
 F-76000 Rouen, France
 samy-melwan.vilhes@insa-rouen.fr

Gilles Gasso
 INSA Rouen, Univ Rouen
 Normandie Univ, LITIS UR 4108
 F-76000 Rouen, France
 gilles.gasso@insa-rouen.fr

Mokhtar Z. Alaya
 Univ de Technologie de Compiègne
 LMAC EA 222
 F-60203 Compiègne, France
 alayaelm@utc.fr

Abstract—Time series anomaly detection (TSAD) focuses on identifying whether observations in streaming data deviate significantly from normal patterns. With the prevalence of connected devices, anomaly detection on time series has become paramount, as it enables real-time monitoring and early detection of irregular behaviors across various application domains. In this work, we introduce PatchTrAD, a Patch-based Transformer model for time series anomaly detection. Our approach leverages a Transformer encoder along with the use of patches under a reconstruction-based framework for anomaly detection. Empirical evaluations on multiple benchmark datasets show that PatchTrAD is on par, in terms of detection performance, with state-of-the-art deep learning models for anomaly detection while being time efficient during inference.

Index Terms—Anomaly detection, Times series, Deep learning, Transformer, Patch

I. INTRODUCTION

Time series anomaly detection (TSAD) refers to the task of identifying whether new observations from a data stream significantly differ from expected normal patterns. Several real-world applications have been considered, including for instance, industrial equipment status surveillance, intrusion detection or home monitoring. The even-increasing scale of sensing-technologies and their widespread in several application domains require efficient and accurate anomaly detection techniques to ensure security. The different types, dimensionality or properties of times series has led to various anomaly detection methods for times series, including deep learning-based approaches [1]–[9].

Many approaches for TSAD under unsupervised learning framework have been proposed. Mainly, they can be categorized as reconstruction-based [3], [5], [6], density-based or level set-based [1], [2], [10], contrastive learning [7]–[9] or prediction-based approaches [11], [12]. Reconstruction models aim to learn a latent representation of the data from which the original samples are reconstructed. A high reconstruction error may be indicative of an anomaly. Transformer encoder-decoder architectures [6] are representatives of this

category of algorithms with promising detection performances. Density/level set-based methods typically perform density or level-set estimation from some latent representation of the time series and predict the likelihood or the score of new observations to be normal. Contrastive learning has been leveraged for TSAD and recently a multiscale patch-based deep architecture [9] that hinges on times series patch-mixing strategy to learn representation adapted to anomaly detection has been introduced. Finally prediction-based approaches rely on recurrent cells such as LSTM or Transformer-based deep architectures, including those using patches [12], to train time-series forecasting models. An anomaly is deemed occurring when the forecasting error for given sequential new samples exceeds a certain threshold, indicating a significant change in the time series.

Building on the effectiveness of patch-based Transformer models for time series forecasting, the lightweight model achieved through patch construction and the efficiency of reconstruction-based methods for TSAD, we propose herein PatchTrAD a model that leverages these approaches to enhance anomaly detection. We show that our patch-based transformer model focusing on reconstruction error leads to state-of-the-art results on both univariate and multivariate time series while remaining fast during inference.

II. RELATED WORKS

A. Preliminary

We formulate the problem as follows: let $x_{1:t} = (x_1, x_2, \dots, x_t)$ denote a stream of data, where an observation at time t consists of M modalities ($x_t \in \mathbb{R}^M$; $M = 1$ for univariate time series, $M \geq 2$ for multivariate time series). The objective is to determine whether the next observation, x_{t+1} is normal or anomalous. In practice, one uses a sliding window of a predefined size w i.e., one relies on the the most recent w observations $x_{t-w+1:t}$ to infer the normality of x_{t+1} .

B. Prediction error-based anomaly detection

A category of techniques to detect anomaly in time series involves training prediction models. These models are trained

using the samples $x_{t-w+1:t}$ to predict x_{t+1} . If the prediction error exceeds a predefined threshold, x_{t+1} is deemed anomalous otherwise, it is considered normal. Typical models include LSTM-based model, Transformer-based model [13]. PatchTST is a Transformer-based model [12] that utilizes patches along with the RevIN [14] invertible normalization technique for handling multivariate time series.

C. Reconstruction error-based anomaly detection

Another class of techniques are reconstruction models, which aim to reconstruct the input window. These models commonly learn a latent representation space in an auto-encoder manner based on windowed inputs $x_{t-w+1:t+1}, \forall t \in [w, \dots, T]$. At inference stage the model projects the input window $x_{t-w+1:t+1}, \forall t > T$ into a latent space and reconstructs it back. Similarly to prediction models, if the reconstruction error exceeds a preset threshold, x_{t+1} is classified as anomalous. Example models are LSTM-based autoencoder [15], MAD-GAN [5] a GAN-based multivariate time series model, USAD [3] a multivariate model with two autoencoders sharing the same encoder trained in adversarial way and TranAD [6] a Transformer-based network that reconstructs the input window using a focus score-based self-conditioning.

D. Other methods

Other methods aim to determine a conformity score, for instance models relied on discrepancy in latent spaces (Anomaly Transformer [7], DCdetecor [8], PatchAD [9]). The Deep One-Class Classifier [1] simultaneously learns a lower-dimensional representation of normal windows and a one-class classifier that estimates the minimum volume data-enclosing hypersphere. Test input windows lying outside the learned hypersphere is deemed abnormal. The Deep Robust One-Class Classifier (DROCC) [2] assumes that the typical training samples lie on a low dimensional locally linear manifold. DROCC employs a gradient ascent step to generate realistic anomalous samples, providing access to the negative class to enhance the anomaly detection.

III. PATCHTRAD

In this work, we propose **PatchTrAD**, a transformer-based reconstruction model that leverages patching techniques for TSAD and focuses on patch-wise reconstruction error. It is inspired from the time series forecasting model PatchTST [12]. An overview of PatchTrAD is detailed in Fig. 1. We adopt the concept of patches similar to the notion of tokens. Namely, patches/tokens are widely used in transformer architectures for vision and natural language processing (e.g. ViT [16], BERT [17]), and are crucial when dealing with local semantic information. Patching for TSAD has been previously explored in [8] and [9]. We further incorporate the concept of channel independence, where each patch contains information from a single modality $m \in \{1, \dots, M\}$.

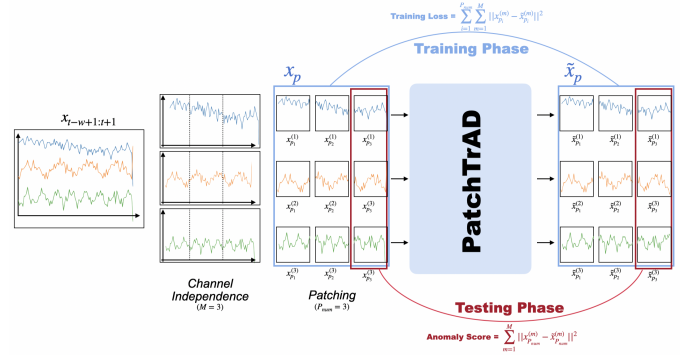


Fig. 1. PatchTrAD Overview.

A. Patching

The input of PatchTrAD is a window $x_{t-w+1:t+1} \in \mathbb{R}^{M \times (w+1)}$. For a given stream of a modality m , denoted as $x_{t-w+1:t+1}^{(m)}$, its patch transformation is determined by the patch length P_{len} and the stride S . Before patching, we pad the m -th stream by repeating S times the last/test observation $x_{t+1}^{(m)}$. Hence, patches can overlap and $x_{t+1}^{(m)}$ belongs to the last patch. The number of patches is given by:

$$P_{\text{num}} = \left\lfloor \frac{(w+1 - P_{\text{len}})}{S} \right\rfloor + 2.$$

Thus, we transform the input window $x_{t-w+1:t+1} \in \mathbb{R}^{M \times (w+1)}$ into the tensor $x_p \in \mathbb{R}^{M \times P_{\text{num}} \times P_{\text{len}}}$. We denote by $x_p^{(m)} \in \mathbb{R}^{P_{\text{num}} \times P_{\text{len}}}$ the set of patches for the m -th modality extracted from x_p and $x_{p_i}^{(m)} \in \mathbb{R}^{P_{\text{len}}}$ represents the i -th patch for the m -th modality, with $i \in \{1, \dots, P_{\text{num}}\}$.

B. Channel independence

Channel independence, in contrast to channel dependency, refers to the scenario where each input patch contains information from only a single modality. Empirical evidence suggest that this setting does not harm performance [12], [18]. In our approach, these input patches are fed into the same Transformer encoder, regardless of their modality. To make our notations more readable, given a tensor $y \in \mathbb{R}^{M \times N \times O}$ and a matrix $W \in \mathbb{R}^{O \times D}$, the tensor-matrix product yW is computed by first flattening y into $y \in \mathbb{R}^{(MN) \times O}$, performing the multiplication to obtain $yW \in \mathbb{R}^{(MN) \times D}$, and then reshaping the result back to $yW \in \mathbb{R}^{M \times N \times D}$. This clarification also applies to tensor-matrix addition.

C. Vanilla transformer encoder layer

Considering channel independence, we set a Vanilla Transformer Encoder [13] that covers multiple layers of residual multi-head self-attention blocks with GELU activation, dropout, and batch normalization (omitted in the equations below). Note that the time dimension is represented by P_{num} .

In a first step, we project x_p using a learnable $W_{\text{proj}} \in \mathbb{R}^{P_{\text{len}} \times D_{\text{model}}}$ and add a fixed positional encoding $W_{\text{pe}} \in \mathbb{R}^{P_{\text{num}} \times D_{\text{model}}}$, where D_{model} denotes the model dimension.

$$\tilde{x}_p = x_p W_{\text{proj}} + W_{\text{pe}} \in \mathbb{R}^{M \times P_{\text{num}} \times D_{\text{model}}}.$$

The single-head attention block for one layer is defined by $W_Q \in \mathbb{R}^{D_{\text{model}} \times D_k}$, $W_K \in \mathbb{R}^{D_{\text{model}} \times D_k}$, $W_V \in \mathbb{R}^{D_{\text{model}} \times D_v}$ and $W_{\text{out}} \in \mathbb{R}^{D_v \times D_{\text{model}}}$ (only one head and one layer presented, with D_k and D_v hidden dimensions). Then:

$$\begin{aligned} Q &= \tilde{x}_p W_Q \in \mathbb{R}^{M \times P_{\text{num}} \times D_k}, \\ K &= \tilde{x}_p W_K \in \mathbb{R}^{M \times P_{\text{num}} \times D_k}, \\ V &= \tilde{x}_p W_V \in \mathbb{R}^{M \times P_{\text{num}} \times D_v}, \\ h &= \text{Softmax} \left(\frac{QK^\top}{\sqrt{D_{\text{model}}}} \right) V \in \mathbb{R}^{M \times P_{\text{num}} \times D_v}, \\ z &= h W_{\text{out}} \in \mathbb{R}^{M \times P_{\text{num}} \times D_{\text{model}}}. \end{aligned}$$

D. Patch head

From here, each modality has its own transformation. The patch head takes as input the output of the encoder $z \in \mathbb{R}^{M \times P_{\text{num}} \times D_{\text{model}}}$. It projects each $z^{(m)} \in \mathbb{R}^{P_{\text{num}} \times D_{\text{model}}}$ back to the patch length size using M learnable linear functions $W_{\text{out}}^m \in \mathbb{R}^{D_{\text{model}} \times P_{\text{len}}}$. Therefore, we have:

$$\begin{aligned} \tilde{x}_p^{(m)} &= z^{(m)} W_{\text{out}}^m \in \mathbb{R}^{P_{\text{num}} \times P_{\text{len}}}, \\ \tilde{x}_p &= \text{concat}(\tilde{x}_p^{(1)}, \dots, \tilde{x}_p^{(M)}) \in \mathbb{R}^{M \times P_{\text{num}} \times P_{\text{len}}}. \end{aligned}$$

A key difference from PatchTST [12] resides in this last layer: instead of flatten heads as in PatchTST, our approach focuses solely on reconstructing the input patches.

E. Training and detection

Training PatchTST [12] leads to compute the MSE loss to compare forecasted values with the ground truth. However PatchTrAD is designed to accurately reconstruct the entire input patch x_p . Thus, the training loss function we consider is the sum squared error between x_p and its reconstruction \tilde{x}_p .

$$\text{training loss} = \sum_{i=1}^{P_{\text{num}}} \sum_{m=1}^M \|x_{p_i}^{(m)} - \tilde{x}_{p_i}^{(m)}\|^2.$$

The patching setting of PatchTrAD ensures that the test observation x_{t+1} is always included in the final patch. During detection phase, the anomaly score is computed through the reconstruction error of the last patch $x_{P_{\text{num}}}$, as it focuses on the final observation—the one under evaluation:

$$\text{anomaly score} = \sum_{m=1}^M \|x_{P_{\text{num}}}^{(m)} - \tilde{x}_{P_{\text{num}}}^{(m)}\|^2.$$

A higher anomaly score implies a greater likelihood that the test observation is anomalous according to our model.

IV. EXPERIMENTS

A. Datasets

To compare PatchTrAD to the state-of-the-art models, we conduct experiments on several datasets, being univariate and multivariate time series. For each dataset, the training set is only composed of normal observations while the test set contains normal and anomalous observations.

In the univariate case, we consider two datasets: *NYC taxi*

demand dataset (0.11% anomalies in test set, $M=1$) and *CPU usage* data from an Amazon's server in a datacenter (0.15%, $M=1$). Both datasets are taken from Numenta Anomaly Benchmark (NAB) [19]. For the multivariate case, we consider several datasets: *Secure Water Treatment (SWaT) Dataset*¹ (12%, $M=51$); *Server Machine Dataset*² (4%, $M=38$); and two NASA datasets: *Mars Science Laboratory (MSL)* satellite dataset (10%, $M=55$) and *Soil Moisture Active Passive (SMAP)* rover dataset³ (13%, $M=25$). SMD, SMAP and MSL are composed of several sub-datasets, We evaluate each model on every sub-dataset and average the performance.

B. Evaluation method

Most prior works on deep learning for TSAD do not rely on the ROC-AUC score, despite its effectiveness in comparing models on various datasets with different class imbalance [20]. Instead, they primarily report F1-Score, Precision, and Recall, after using *Point Adjustment* (PA) method used for the first time in [4]. However, these metrics require setting a threshold, but this choice depends on the application.

It is worth to note that PA algorithm modifies the model's detections using ground-truth labels before evaluation. Specifically, it considers an entire anomaly period as correctly detected if the model identifies at least one anomaly within that period. PA improves significantly the model's performance, to the point where even a random model can achieve strong detection performance (measured for instance by a F1 score). A detailed study challenging this method is in [21].

To ensure a fair and easily interpretable comparison, we rely solely on ROC-AUC score without applying PA. This evaluation scheme eliminates the need to determine a threshold for model comparison, which is implicitly handled within ROC-AUC metric.

C. Pre-processing and hyperparameters

We normalize each modality of the time series using statistics computed from the training set. This ensures consistency across all models, as they share the same preprocessing steps. Considering hyperparameters, we use the same batch size and window size for each model, adjusting them based on the dataset. For PatchTrAD, we use a patch size of 8 and a stride of 6 each time. Thus, patches overlap and, by construction, the observation to test is on the last patch. We replicate original implementations from the authors' GitHub repositories. When necessary, we make slight modifications to the models dimension to ensure they fit within a single GPU (NVIDIA RTX 2000 Ada Generation Laptop GPU). This adjustment is crucial, as we focus on real-time applications where very large models may be impractical for continuous deployment in production environments.

¹Credited to iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design.

²Credited to the Tsinghua Netman Lab: <https://github.com/NetManAI/Ops/OmniAnomaly>

³Credited to the NASA Jet Propulsion Laboratory: <https://github.com/khundman/telemanom>

TABLE I
ROC-AUC SCORES (**BOLD**: FIRST, UNDERLINE: SECOND, *italic*: THIRD)

Dataset	NYC-Taxi	EC2	MSL	SWaT	SMAP	SMD	Mean	Rank
Model	ROC-AUC							
DC-Detector	0.498	0.827	0.536	0.435	0.560	0.530	0.564	11.8
DROCC	0.529	0.886	0.531	0.751	0.569	0.638	0.651	11.0
MADGAN	0.782	0.011	0.499	0.791	0.544	0.708	0.556	10.0
USAD	0.675	<i>0.977</i>	<u>0.622</u>	0.814	0.448	0.638	0.696	8.8
PatchTST-rev ^a	0.552	0.999	0.562	0.233	0.498	<i>0.873</i>	0.620	8.7
DOC	0.704	0.804	0.603	0.404	0.583	0.766	0.644	8.5
LSTM-rev ^a	0.646	0.998	0.598	0.238	0.520	0.858	0.643	8.4
AnomalyTransformer	0.491	0.994	<i>0.609</i>	0.819	<u>0.637</u>	0.678	0.705	7.7
LSTM	0.511	0.999	0.582	<i>0.842</i>	0.604	0.833	0.729	6.5
AE-LSTM	0.664	0.998	0.589	0.840	0.614	0.828	<i>0.756</i>	6.0
PatchTST	0.696	0.999	0.560	<u>0.843</u>	0.514	<u>0.882</u>	0.749	5.5
TranAD	0.551	0.967	<u>0.622</u>	0.815	0.668	0.884	0.751	5.2
PatchAD	0.972	0.998	0.625	0.822	<i>0.630</i>	0.818	<u>0.811</u>	<u>4.1</u>
PatchTrAD (ours)	<u>0.922</u>	0.999	<u>0.622</u>	0.845	0.629	0.869	0.814	2.8

^aRevin normalization applied.

D. Results

As shown in Table I, PatchTrAD competes with the top-performing models. It achieves the best performance according to its rank and overall mean performance. Both PatchTrAD and PatchAD leverage the patching technique. However, PatchTrAD is a reconstruction-based model using attention, whereas PatchAD is a discrepancy-based model without attention. Another top competitor is TranAD, which is also a reconstruction-based model with attention but does not incorporate patching. TranAD excels on multivariate datasets but performs less effectively than PatchTrAD and PatchAD on univariate datasets.

We rank all methods using the post-hoc Nemenyi test [22]. The diagram in Fig. 2 serves not as a definitive conclusion but as one from several ways to describe the performance of the predictors. According to this test, PatchTrAD ranks first,

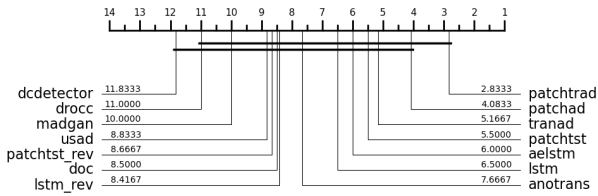


Fig. 2. Critical difference diagrams for AUC scores using the post-hoc Nemenyi test with $\alpha = 5\%$, where better-ranked methods appear on the upper right.

followed by PatchAD in second place. TranAD achieves a better mean rank than the LSTM-based AutoEncoder (while TranAD achieves a lower overall mean AUC). Additionally, we identify distinct groups of models with significantly different performance levels (bold lines). The first group includes all models except PatchTrAD, indicating no significant difference among them. Since PatchTrAD ranks first, this suggests that it is the best-performing model according to this test. Conversely,

the second group consists of all models except DCDetector, the worst-performing model. This suggests that all models perform similarly, except for DCDetector, which is noticeably less effective. The implementation of PatchTrAD is available at: <https://github.com/vilhess/PatchTrAD>

E. Inference-time computation

As previously concluded, three models stand out: PatchTrAD (ours), PatchAD, and TranAD. Since we focus on real-time anomaly detection, the models under consideration should be both fast and efficient during inference. In Fig. 3, we depict inference times of these models according to w , the size of the time window. As it can be noticed, PatchAD is by far the most time-consuming and PatchTrAD is more efficient than PatchAD. However, PatchTrAD is still behind TranAD, and this gap becomes more noticeable as the window size increases.

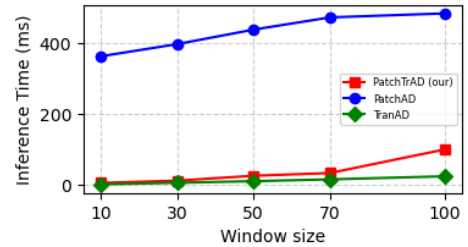


Fig. 3. Inference-time Computation based on SWaT Dataset configuration for various window sizes, with a batch size of 128.

V. CONCLUSION

We introduced PatchTrAD, a transformer-based model leveraging patches for anomaly detection focusing on reconstruction error. This model competes with state-of-the-art approaches and is almost 3 times faster than the best-competitor model PatchAD. It performs well across diverse

datasets and remains efficient during inference, making it suitable to a wide range of univariate and multivariate time series. We hence believe that PatchTrAD might be strong potential for further industrial TSAD problems. Future work could explore pretraining the transformer encoder on a diverse range of time series, followed by fine-tuning the patch head for each new time series, as this approach would improve generalization and enable efficient transfer learning.

VI. APPENDICES

A. Ablation study: patch size and stride impact

PatchTrAD’s architecture is determined by the patch size and stride, which together define the number of patches. In this section, we analyze how these parameters impact the final score by evaluating the model exclusively on NYC Taxi Demand and SWaT datasets.

TABLE II
ROC-AUC OF PATCHTRAD WITH VARYING STRIDES AND PATCH SIZES
(**BOLD**: FIRST, UNDERLINE: SECOND)

Dataset	NYC Taxi ($w = 32$)	SWaT ($w = 100$)
P_{len} S	ROC-AUC	
3 3	0.776	0.839
5 3	0.904	0.839
5 5	0.832	0.839
6 6	0.872	0.842
8 3	0.838	0.846
8 5	0.801	0.844
8 6	0.922	<u>0.845</u>
8 8	<u>0.917</u>	<u>0.845</u>
16 12	0.536	0.821
16 16	0.801	0.820
28 22	0.890	0.822
28 28	0.544	0.823
32 28	0.549	0.829
32 32	0.568	0.825

We observe in Table II that for PatchTrAD to perform optimally, it’s crucial to find the right balance. If the patch size and stride are too large, performance decreases. Conversely, if they are too small, the model does not achieve its best results. Based on our experiments, a patch length of 8 and a stride of 6 yield the best detection performances. We do not consider strides greater than the patch length, as this would result in not considering all observations within the window.

REFERENCES

- [1] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, “Deep one-class classification,” in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 4393–4402, PMLR, 10–15 Jul 2018.
- [2] S. Goyal, A. Raghunathan, M. Jain, H. Simhadri, and P. Jain, “Drocc: deep robust one-class classification,” in *Proceedings of the 37th International Conference on Machine Learning*, ICML’20, JMLR.org, 2020.
- [3] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, “Usad: Unsupervised anomaly detection on multivariate time series,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD ’20, (New York, NY, USA), p. 3395–3404, Association for Computing Machinery, 2020.
- [4] 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*, WWW ’18, (Republic and Canton of Geneva, CHE), p. 187–196, International World Wide Web Conferences Steering Committee, 2018.
- [5] 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,” in *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series* (I. V. Tetko, V. Kůrková, P. Karpov, and F. Theis, eds.), (Cham).
- [6] S. Tuli, G. Casale, and N. R. Jennings, “Tranad: deep transformer networks for anomaly detection in multivariate time series data,” *Proc. VLDB Endow.*, vol. 15, p. 1201–1214, Feb. 2022.
- [7] J. Xu, H. Wu, J. Wang, and M. Long, “Anomaly transformer: Time series anomaly detection with association discrepancy,” in *International Conference on Learning Representations*, 2022.
- [8] Y. Yang, C. Zhang, T. Zhou, Q. Wen, and L. Sun, “Dedector: Dual attention contrastive representation learning for time series anomaly detection,” in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD ’23, (New York, NY, USA), p. 3033–3045, Association for Computing Machinery, 2023.
- [9] Z. Zhong, Z. Yu, Y. Yang, W. Wang, and K. Yang, “Patchad: A lightweight patch-based mlp-mixer for time series anomaly detection,” 2024.
- [10] 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.
- [11] 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*, 2023.
- [12] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, “A time series is worth 64 words: Long-term forecasting with transformers,” in *The Eleventh International Conference on Learning Representations*, 2023.
- [13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS’17, (Red Hook, NY, USA), p. 6000–6010, Curran Associates Inc., 2017.
- [14] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo, “Reversible instance normalization for accurate time-series forecasting against distribution shift,” in *International Conference on Learning Representations*, 2022.
- [15] O. I. Provotar, Y. M. Linder, and M. M. Veres, “Unsupervised anomaly detection in time series using lstm-based autoencoders,” in *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*, pp. 513–517, 2019.
- [16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” in *International Conference on Learning Representations*, 2021.
- [17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *NAACL (J. Burstein, C. Doran, and T. Solorio, eds.)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.
- [18] L. Han, H.-J. Ye, and D.-C. Zhan, “The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting,” 2023.
- [19] A. Lavin and S. Ahmad, “Evaluating real-time anomaly detection algorithms – the numanta anomaly benchmark,” in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, IEEE, Dec. 2015.
- [20] E. Richardson, R. Trevizani, J. A. Greenbaum, H. Carter, M. Nielsen, and B. Peters, “The receiver operating characteristic curve accurately assesses imbalanced datasets,” *Patterns*, vol. 5, no. 6, p. 100994, 2024.
- [21] K. Siwon, K. Choi, H.-S. Choi, B. Lee, and S. Yoon, “Towards a rigorous evaluation of time-series anomaly detection,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 7194–7201, 06 2022.
- [22] J. Demšar, “Statistical comparisons of classifiers over multiple data sets,” *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1–30, 2006.