Supplementary Material: Nominality Score Conditioned Time Series Anomaly Detection by Point/Sequential Reconstruction

Anonymous Author(s)
Affiliation
Address
email

A The Best F1 Score

This section describes the method for calculating the best F1 score ($\mathrm{F1}^*$) from a set of anomaly scores $\mathbf{a} = \{a_1, ..., a_T\}$ and a set of labels $\mathbf{y} = \{y_1, ..., y_T\}$. Firstly, given $\mathbf{a}$ and some arbitrary threshold θ_a, we can calculate $\hat{\mathbf{y}} = \{\hat{y}_1, ...\hat{y}_T\}$, where $\hat{y}_t \triangleq \mathbb{1}_{a_t \geq \theta_a}$. Secondly, $\hat{\mathbf{y}}$ is used to calculate TP, FP, and FN, which corresponds to the sets of time points for true positives, false positives, and false negatives.

$$\mathrm{TP} \triangleq \{t|\hat{y}_t = 1, y_t = 1\}, \;\; \mathrm{FP} \triangleq \{t|\hat{y}_t = 1, y_t = 0\}, \;\; \mathrm{FN} \triangleq \{t|\hat{y}_t = 0, y_t = 1\} \tag{1}$$

Thirdly, we calculate the precision (P) and recall (R), and then calculate the F1 score, which is the harmonic mean between R and P.

$$\mathrm{F1} \triangleq \frac{2\mathrm{PR}}{\mathrm{P} + \mathrm{R}}, \;\; \mathrm{P} \triangleq \frac{\mathrm{n(TP)}}{\mathrm{n(TP)} + \mathrm{n(FP)}}, \;\; \mathrm{R} \triangleq \frac{\mathrm{n(TP)}}{\mathrm{n(TP)} + \mathrm{n(FN)}} \tag{2}$$

Finally, we calculate $\mathrm{F1}^*$ by using the threshold that yields the highest F1.

$$\mathrm{F1}^*(\mathbf{a}; \mathbf{y}) \triangleq \max_{\theta_a} \mathrm{F1}(\hat{\mathbf{y}}(\mathbf{a}, \theta_a); \mathbf{y}) \tag{3}$$

B Model Architecture

Performers (an improved variant of Transformers) are competitive in terms of execution speed compared with other Transformer variants [1, 2], hence we use them as the basic building block of our models. For both $\mathcal{M}_{pt}$ and $\mathcal{M}_{seq}$, we use a linear layer with input and output dimensions equal to D as the token embedding layer, a fixed positional embedding layer at the beginning, a feature redraw interval of 1, and a tanh activation function immediately before the output. GELUs [3] are used as the activation layer for all linear layers. We do not change any other predefined activation layer inside Performers.

B.1 Performer-based autoencoder

For the Performer-based autoencoder $\mathcal{M}_{pt}$, the input with the shape $(batch, W, D)$ is passed through a plain Performer with N_{perf} layers after the positional embedding step, where W represents the input window size. This is followed by a linear encoding layer that transforms the dimensionality from D to D_{lat}, resulting in the shape $(batch, W, D_{lat})$ for the latent variables. In the case of $\mathcal{M}_{pt}$, there is no compression along the time domain. The latent variables then pass through another linear decoding layer that transforms the dimensionality from D_{lat} back to D, followed by another Performer with N_{perf} layers.

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

B.2 Performer-based stacked encoder

For the Performer-based stacked encoder $\mathcal{M}_{seq}$, the input with shape $(batch, W_0, D)$ is passed through a plain Performer with one layer after the positional embedding step, followed by a linear encoding layer that transforms the window size from W_0 to W_1, where $W_0 = 2\gamma$ (cf. section 2.6). It should be noted that unlike $\mathcal{M}_{pt}$, compression is done along the time domain for $\mathcal{M}_{seq}$. The one-layer Performer and linear layer are stacked N_{enc} times, where in the i-th stack, the window size is compressed from W_{i-1} to W_i. $W_{N_{enc}}$ is equal to the target output window size δ. For both $\mathcal{M}_{pt}$ and $\mathcal{M}_{seq}$, we optimize N_{perf}, D_{lat}, W, N_{enc}, W_i for $i \in \{0, ..., N_{enc}\}$, and δ for the best performance. Note that $\mathcal{M}_{seq}$ isn't capable of reconstructing the first and last γ time points due to its architecture, hence we discard the first and last γ points reconstructed by $\mathcal{M}_{pt}$ so that rest of the time points have exactly two reconstructed values corresponding to using $\mathcal{M}_{seq}$ and $\mathcal{M}_{pt}$, respectively.

C Data Preprocessing and Training Details

Table 1 shows the hyperparameters used for implementing NPSR on the experimented datasets. To ensure fair comparison, the same preprocessing method is applied to all algorithms for the same dataset. The search for hyperparameters is done manually, starting from some reasonable value (e.g. a learning rate of 10^{-4}). The authors believe that there is still room for improvement by fine-tuning these hyperparameters. To speed up training, we load all training inputs and outputs, and testing inputs onto the GPU before training. We use a local GPU, which can be either GeForce RTX 3070 (8GB), 3080 (12GB) or 3090 (24GB). For an individual experiment using a single dataset and training method, the training time ranges from approximately 2 minutes to 12 hours. For single-entity datasets and multi-entity datasets that use the combined training method, we run the experiments for at least 3 times and confirm that the results are stable given different random seeds. For multi-entity datasets with entities trained individually, the results are averaged across all entities. Generally, datasets with single entities train faster than those with multiple entities. There are some additional remarks regarding the preprocessing of the datasets. For SWaT, we use `SWaT_Dataset_Attack_v0.csv` and `SWaT_Dataset_Normal_v1.csv` from the folder `SWaT.A1 & A2_Dec 2015` (manually converted from `*.xlsx`). We corrected some original flaws in the dataset (e.g. redundant blank spaces in some labels), and set the 5th and 10th columns to all 0. For WADI, we use the 2017 year dataset. Columns with excessive NaNs (more than half of the entire length) are deleted. Other NaNs are forward-filled. After deleting all the columns with excessive NaNs, the 86th column is further set to all 0. For PSM, we forward-fill all NaNs. For MSL and SMD, two additional blank channels are added to make the number of channels divisible by the number of heads.

Table 1: Implementation details. (c) stands for the combined training method (cf. section 3.2).

Parameter \ Dataset	SWaT	WADI	PSM	MSL	MSL (c)	SMAP	SMAP (c)	SMD	SMD (c)
				Preprocess					
Downsample	10	10	10	1	1	1	1	2	2
Normalization	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax	Minmax
Stride	10	10	10	10	10	10	10	10	10
W (for $\mathcal{M}_{pt}$)	100	100	100	100	100	50	50	50	50
W_0 (for $\mathcal{M}_{seq}$)	100	100	100	50	50	50	50	50	50
δ	20	20	20	6	6	6	6	6	6
clamping (test data)	[-4,4]	[-4,4]	[-4,4]	[-4,4]	[-4,4]	[-4,4]	[-4,4]	[-4,4]	[-4,4]
				Model architecture					
# of heads	9	14	5	11	12	5	10	8	11
D_{lat}	10	10	10	10	10	10	10	10	10
ff_mult	4	4	4	4	4	4	4	4	4
N_{perf}	4	4	4	4	4	4	4	4	4
N_{enc}	8	8	8	8	8	8	8	8	8
				Induced anomaly score					
Gate function	soft	soft	soft	soft	hard	soft	hard	soft	soft
d	16	16	64	128	128	64	64	16	1
Ratio of N_{trn} for θ_N	99.85%	99.85%	99.85%	99.85%	97.5%	99.85%	∞	99.85%	99.85%
				Training					
Learn rate	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Batch size	64	64	64	64	64	64	64	64	64
Training epochs	100	100	80	40	20	50	20	40	10

D The Point-adjusted Best F1 Score

Analogous to $\mathrm{F1}^*$, the point-adjusted best F1 score ($\mathrm{F1}^*_{\mathrm{PA}}$) corresponds to $\mathrm{F1}^*$ calculated after point-adjustment. Table 2 shows the $\mathrm{F1}^*$ and $\mathrm{F1}^*_{\mathrm{PA}}$ of different algorithms, including NPSR, applied to several datasets.

The results suggest that $\mathrm{F1}^*_{\mathrm{PA}}$ may not be reliable - on the SWaT, WADI, PSM, and MSL datasets, simple heuristic approaches (e.g. using the mean squared value of an input time point as the anomaly score) outperform all deep learning methods when evaluated using $\mathrm{F1}^*_{\mathrm{PA}}$. Moreover, optimizing on $\mathrm{F1}^*_{\mathrm{PA}}$ does not necessarily guarantee a higher $\mathrm{F1}^*$. NPSR is optimized on $\mathrm{F1}^*$ by tuning the algorithm-specific parameters (e.g. d) and general parameters. To additionally optimize on $\mathrm{F1}^*_{\mathrm{PA}}$, we simply added *spikes* to the induced anomaly score ($\hat{A}_{spike}(\cdot)$) with value ∞ for some fixed interval s. The results show that $\hat{A}_{spike}(\cdot)$ can also achieve competitive $\mathrm{F1}^*_{\mathrm{PA}}$ values.

Table 2: Point-adjusted best F1 score ($\mathrm{F1}^*_{\mathrm{PA}}$) and best F1 score ($\mathrm{F1}^*$) results on several datasets, with bold text denoting the highest and underlined text denoting the second highest value. The deep learning methods are sorted with older methods at the top and newer ones at the bottom.

Dataset	SWaT		WADI		PSM		MSL		SMAP		SMD	
Metric	$\mathrm{F1}^*_{\mathrm{PA}}$	$\mathrm{F1}^*$	$\mathrm{F1}^*_{\mathrm{PA}}$	$\mathrm{F1}^*$	$\mathrm{F1}^*_{\mathrm{PA}}$	$\mathrm{F1}^*$	$\mathrm{F1}^*_{\mathrm{PA}}$	$\mathrm{F1}^*$	$\mathrm{F1}^*_{\mathrm{PA}}$	$\mathrm{F1}^*$	$\mathrm{F1}^*_{\mathrm{PA}}$	$\mathrm{F1}^*$
Simple Heuristics [4, 5, 6]	**0.969**	0.789	**0.965**	0.353	**0.985**	0.509	**0.965**	0.239	0.961	0.229	0.934	0.494
DAGMM [7]	0.853	0.750	0.209	0.121	0.761	0.483	0.701	0.199	0.712	0.333	0.723	0.238
LSTM-VAE [8]	0.805	0.776	0.380	0.227	0.809	0.455	0.854	0.212	0.756	0.235	0.808	0.435
MSCRED [9]	0.807	0.757	0.374	0.046	0.626	0.556	0.936	0.250	0.866	0.170	0.841	0.382
OmniAnomaly [10]	0.866	0.782	0.417	0.223	0.664	0.452	0.901	0.207	0.854	0.227	<u>0.962</u>	0.474
MAD-GAN [11]	0.815	0.770	0.556	0.370	0.658	0.471	0.917	0.267	0.865	0.175	0.915	0.220
MTAD-GAT [12]	0.860	0.784	0.602	0.437	0.780	0.571	0.908	0.275	0.901	0.296	0.908	0.400
USAD [13]	0.846	0.792	0.430	0.233	0.725	0.479	0.911	0.211	0.819	0.228	0.946	0.426
THOC [14]	0.881	0.612	0.506	0.130	0.895	-	0.937	0.190	0.952	0.240	0.541	0.168
UAE [4]	0.869	0.453	<u>0.957</u>	0.354	0.936	0.427	0.920	<u>0.451</u>	0.896	0.390	**0.972**	0.435
GDN [15]	0.935	<u>0.810</u>	0.855	<u>0.570</u>	0.923	0.552	0.903	0.217	0.708	0.252	0.716	<u>0.529</u>
GTA [16]	0.910	0.761	0.84	0.531	0.855	0.542	0.911	0.218	0.904	0.231	0.919	0.351
Anomaly Transformer [17]	0.941	0.019	0.714	0.015	<u>0.979</u>	0.022	0.936	0.021	<u>0.967</u>	0.019	0.923	0.021
TranAD [18]	0.815	0.669	0.495	0.415	0.882	**0.649**	0.949	0.251	0.892	0.247	0.961	0.310
NPSR (combined)	-	-	-	-	-	-	<u>0.960</u>	0.261	**0.978**	**0.511**	0.850	0.252
NPSR	<u>0.953</u>	**0.839**	0.938	**0.642**	0.957	<u>0.648</u>	-	**0.551**	-	<u>0.437</u>	-	**0.535**

E Source of Data

Table 3 shows the data sources used to produce Table 2, as well as the sources for section 3.3. The reference number is followed by a number between 1 and 3, where 1 indicates that the data comes from the original work, 2 indicates that the data comes from reproduced values from another literature, and 3 indicates that we have reproduced the values using public repositories..

F Broader Impacts

The detection of anomalies in time series data can minimize downtime and avert financial losses. Utilizing real-time monitoring of system conditions, anomaly detection techniques for time series data can automatically detect deviations from the expected system behavior, thereby avoiding potential risks and financial harm. This has the potential to reduce the need for manual monitoring of faults and to expedite decision-making processes. Additionally, it can promote the sustainability of AI by preventing energy wastage and system malfunction.

Table 3: Data sources for algorithms and datasets. **Reproduced by using the squared value of channel 1 as the anomaly score.

Dataset	SWaT		WADI		PSM		MSL		SMAP		SMD	
Metric	$F1^*_{PA}$	$F1^*$	$F1^*_{PA}$	$F1^*$	$F1^*_{PA}$	$F1^*$	$F1^*_{PA}$	$F1^*$	$F1^*_{PA}$	$F1^*$	$F1^*_{PA}$	$F1^*$
Simple Heuristic	[5] - 1	[5] - 1	[5] - 1	[5] - 1	[6] - 1	**	[6] - 1	[5] - 1	[6] - 1	[5] - 1	[6] - 1	[5] - 1
DAGMM	[5] - 2	[18] - 3	[5] - 2	[5] - 2	[18] - 3	[18] - 3	[5] - 2	[5] - 2	[5] - 2	[5] - 2	[5] - 2	[5] - 2
LSTM-VAE	[13] - 2	[13] - 2	[13] - 2	[13] - 2	[17] - 2	[19] - 3	[13] - 2	[5] - 2	[13] - 2	[5] - 2	[13] - 2	[5] - 2
MSCRED	[18] - 2	[18] - 3	[18] - 2	[18] - 3	[18] - 3	[18] - 3	[18] - 2	[18] - 3	[18] - 2	[18] - 3	[18] - 2	[18] - 3
OmniAnomaly	[5] - 2	[5] - 2	[5] - 2	[5] - 2	[18] - 3	[18] - 3	[10] - 1	[5] - 2	[10] - 1	[5] - 2	[10] - 1	[5] - 2
MAD-GAN	[18] - 3	[11] - 1	[18] - 3	[11] - 1	[18] - 3	[18] - 3	[18] - 2	[18] - 3	[18] - 2	[18] - 3	[18] - 2	[18] - 3
MTAD-GAT	[20] - 2	[12] - 3	[20] - 2	[12] - 3	[12] - 3	[12] - 3	[12] - 1	[12] - 3	[12] - 1	[12] - 3	[20] - 2	[12] - 3
USAD	[13] - 1	[13] - 1	[13] - 1	[13] - 1	[18] - 3	[18] - 3	[13] - 1	[5] - 2	[13] - 1	[5] - 2	[13] - 1	[5] - 2
THOC	[14] - 1	[5] - 2	[5] - 2	[5] - 2	[17] - 2	-	[14] - 1	[5] - 2	[14] - 1	[5] - 2	[5] - 2	[5] - 2
UAE	[4] - 1	[4] - 1	[4] - 1	[4] - 1	[4] - 3	[4] - 3	[4] - 1	[4] - 1	[4] - 1	[4] - 1	[4] - 1	[4] - 1
GDN	[5] - 2	[15] - 1	[5] - 2	[15] - 1	[15] - 3	[15] - 3	[5] - 2	[5] - 2	[5] - 2	[5] - 2	[5] - 2	[5] - 2
GTA	[16] - 1	[16] - 3	[16] - 1	[16] - 3	[16] - 3	[16] - 3	[16] - 1	[16] - 3	[16] - 1	[16] - 3	[16] - 3	[16] - 3
AnomalyTransformer	[17] - 1	[17] - 3	[17] - 3	[17] - 3	[17] - 1	[17] - 3	[17] - 1	[17] - 3	[17] - 1	[17] - 3	[17] - 1	[17] - 3
TranAD	[18] - 1	[18] - 3	[18] - 1	[18] - 3	[18] - 3	[18] - 3	[18] - 1	[18] - 3	[18] - 1	[18] - 3	[18] - 1	[18] - 3

G Limitations

Despite exhibiting competitive performance against other models, the proposed NPSR algorithm has a few limitations. The point-based model used in training does not incorporate temporal information, which makes it challenging to effectively reconstruct low-dimensional datasets. This issue is particularly challenging for univariate time series since raw inputs would not work for point-based models. One possible solution to this problem is to increase dimensionality by aggregating multiple time points. However, the effectiveness of this approach is yet to be confirmed.

Another limitation of NPSR is the absence of an automatic threshold-finding method, which makes it difficult to determine a suitable threshold when deploying the model. To address this issue, one can define a target false positive rate and estimate the threshold that achieves this target rate using the validation set since only normal data is needed.

References

[1] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[2] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

[3] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[4] Astha Garg, Wenyu Zhang, Jules Samaran, Ramasamy Savitha, and Chuan-Sheng Foo. An evaluation of anomaly detection and diagnosis in multivariate time series. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2508–2517, jun 2022.

[5] Siwon Kim, Kukjin Choi, Hyun-Soo Choi, Byunghan Lee, and Sungroh Yoon. Towards a rigorous evaluation of time-series anomaly detection, 2022.

[6] Keval Doshi, Shatha Abudalou, and Yasin Yilmaz. Reward once, penalize once: Rectifying time series anomaly detection. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022.

[7] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.

[8] Daehyung Park, Yuuna Hoshi, and Charles C Kemp. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters*, 3(3):1544–1551, 2018.

[9] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1409–1416, 2019.

[10] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan 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 '19, page 2828–2837, New York, NY, USA, 2019. Association for Computing Machinery.

[11] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong 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: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part IV*, pages 703–716. Springer, 2019.

[12] Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. Multivariate time-series anomaly detection via graph attention network. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 841–850. IEEE, 2020.

[13] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria 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, page 3395–3404, New York, NY, USA, 2020. Association for Computing Machinery.

[14] Lifeng Shen, Zhuocong Li, and James Kwok. Timeseries anomaly detection using temporal hierarchical one-class network. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13016–13026. Curran Associates, Inc., 2020.

[15] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4027–4035, May 2021.

[16] Zekai Chen, Dingshuo Chen, Xiao Zhang, Zixuan Yuan, and Xiuzhen Cheng. Learning graph structures with transformer for multivariate time-series anomaly detection in iot. *IEEE Internet of Things Journal*, 9(12):9179–9189, 2021.

[17] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series anomaly detection with association discrepancy. *arXiv preprint arXiv:2110.02642*, 2021.

[18] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. Tranad: Deep transformer networks for anomaly detection in multivariate time series data. *arXiv preprint arXiv:2201.07284*, 2022.

[19] TimyadNyda. Github - timyadnyda/variational-lstm-autoencoder: Lstm variational auto-encoder for time series anomaly detection and features extraction.

[20] Jun Zhan, Siqi Wang, Xiandong Ma, Chengkun Wu, Canqun Yang, Detian Zeng, and Shilin Wang. Stgat-mad : Spatial-temporal graph attention network for multivariate time series anomaly detection. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3568–3572, 2022.