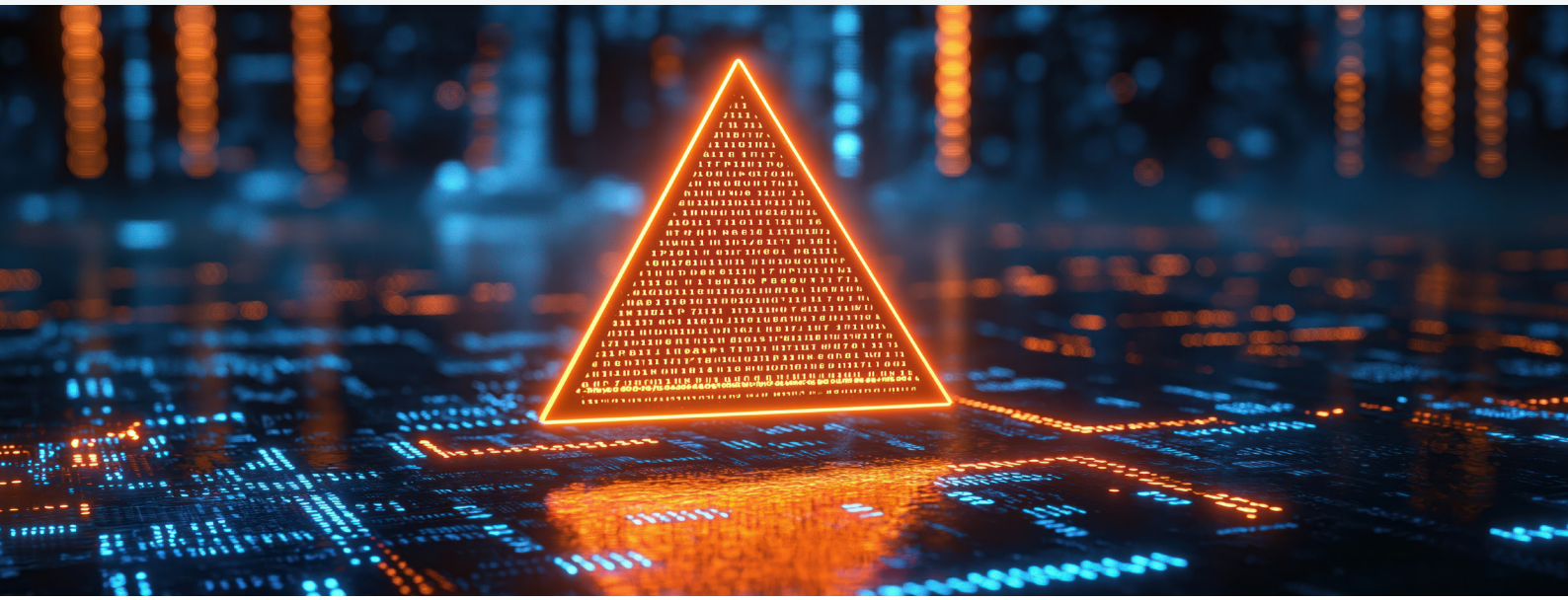
ICT GROUP

Identifying Anomalous Behavior using the Anomaly Surfer

Anomaly detection plays a critical role in various industries, where the presence of abnormal behavior or unexpected patterns can indicate potential problems or opportunities. For example, in the manufacturing industry, anomaly detection can help identify equipment malfunctions and prevent downtime. However, the accurate detection of anomalies is difficult for a number of reasons, such as the stochastic nature of sensor data, high data dimensionality, and the difficulty in defining what is normal and what is abnormal behavior. In this paper, we introduce a technology developed to increase anomaly detection accuracy on time series data: the Anomaly Surfer, a light-weight deep learning anomaly detection model based on the wavelet transform and maximum likelihood estimation. We show that our model performs comparatively better than other state-ofthe-art models by benchmarking it on real-life and synthetic datasets often used in research. Furthermore, we identify multiple practical use cases in which the Anomaly Surfer technology could be of great value.

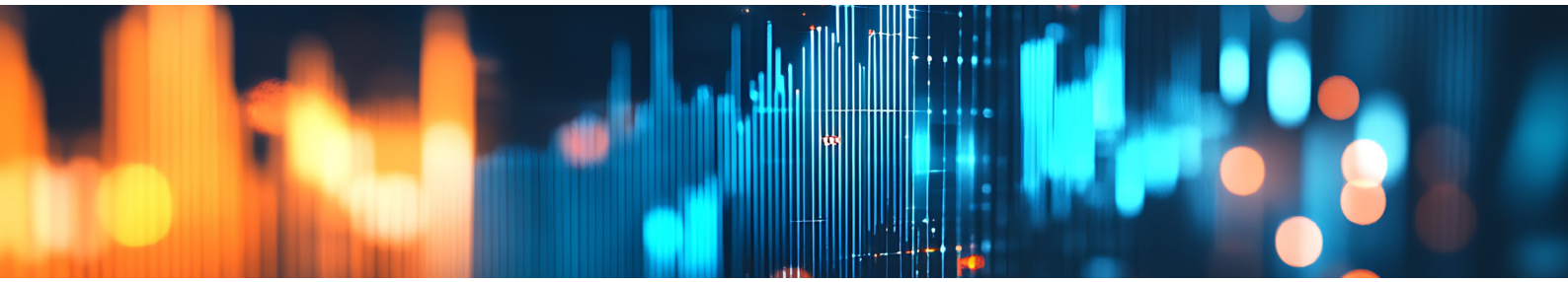
Correspondence: centerofexcellence@ict.nl



1. Introduction

In a world that is ever more connected and focused on data, time series data is prevalent, extending across sectors such as finance, e-commerce, industry, and utilities. Detecting anomalies within time series data is a crucial process that helps to detect potential complications, alleviate risks, and seize emerging prospects. Anomaly detection in time series data is vital in practices like fraud detection, predictive maintenance, network intrusion detection, patient health monitoring, and quality control.

Failing to detect anomalies in time can have drastic consequences, resulting in financial burdens, operational interference, or even devastating failures. In the financial domain, for instance, anomalies can mark dishonest activities or marketplace inconsistencies. In the industrial sector, they could symbolize equipment malfunctions or quality concerns. Early detection and prompt action on these anomalies can assist organizations in risk reduction, efficiency improvement, and maintenance of a competitive lead.



2. The Problem

Detecting anomalies in time series data is hard for a multitude of reasons. First of all, determining what behavior is normal is not a trivial task; the definition of an “anomaly” can vary heavily depending on context or domain. Moreover, there is a lot of variability when it comes to what type of anomalies there are, making it hard to find solutions that work well for all of them.

Another prevalent problem arises when we consider that sensor data is often stochastic in nature. This stochasticity (or randomness) can make it hard to distinguish actual anomalies from noise or other induced variability. For example, in some instances, noise might very well indicate the actual anomaly, whereas in other instances it might obscure it altogether. How to deal with noise remains a difficult problem that is hard to generalize.

Finally, there are some other issues that frequently pop up when we consider anomaly detection in real-life situations. In some multivariate situations, where we have multiple data series, anomaly detection can become hard due to high data dimensionality, i.e., the more features that are included in the data, the more difficult it can become to determine relationships and discern anomalies. Moreover, anomalies often occur only sporadically, which leads to imbalanced training datasets. This makes it hard for models to accurately detect anomalies.

2.1 Types of anomalies

Whilst anomalies can occur in many different varieties, we can generally place them within one of the following three categories:

1. **Point anomalies:** Are often individual or a very small group of values that are discrepant with respect to the behavior and patterns of the surrounding data. These anomalies are generally easy to discern due to them often occurring abruptly or distinctly in data series, such as shown in *Figure 1*.

2. **Collective anomalies:** Are anomalies that generally occur as a larger consecutive serie of data points that is anomalous compared to the entire dataset. These anomalies might be hard to find if we were to zoom in on the dataset, but stand out if we look at the series overall, such as in *Figure 2*.

3. **Contextual anomalies:** These type of anomalies are notoriously difficult to detect, as they constitute behavior that is anomalous with respect to the context from which the data originates, but not necessarily anomalous on a statistical or behavioral level. For example, a sudden drop in electricity usage in households might be normal during the night, but would be anomalous during evening hours.

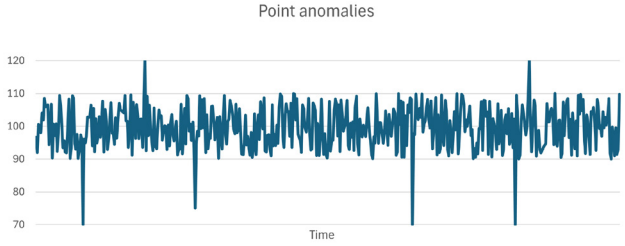


Figure 1: Example of point anomalies.

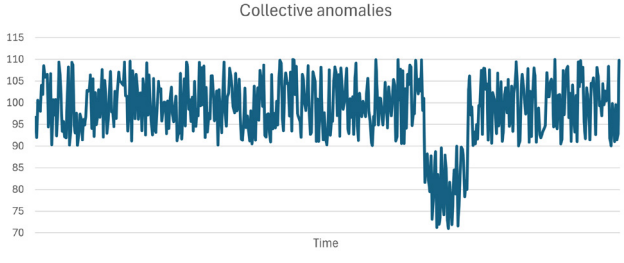


Figure 2: Example of a collective anomaly.

2.2 Anomaly detection models

There are many ways to go about anomaly detection. Traditional approaches typically employ signal processing and statistical analysis techniques, with manual input from specialists to establish triggers and thresholds. Due to the high costs of storage and computing, these methods usually rely on extracting statistical, spectral, or temporal features. However, with the increasing complexity of processes and machines, alongside the rise in data volume from Industry 4.0, many of these methods have become insufficient. Classical techniques like Principal Component Analysis and Fourier transform, though scalable and simple, lack the flexibility and accuracy required for contemporary applications. Improved computational capabilities have paved the way for machine learning and deep learning methods –with the latter gaining popularity for its ability to comprehend complex patterns without making assumptions about the data. One thing to consider when it comes to choosing deep learning models is whether the model is supervised, unsupervised, or semi-supervised. While supervised models usually deliver better results, they depend on labeled input data, which is often unavailable in industrial settings. Hence, most models that can be applied in general settings, like our model, are typically unsupervised or semi-supervised.

Although deep learning models have been shown to be very flexible and powerful in detecting anomalies, they generally suffer from a number of drawbacks:

- **Generalization:** Despite achieving good results, many models are tailored specifically towards one task or use case.
- **Scalability:** Many DL models have trouble when encountering higher data dimensionality, complex temporal patterns and high data volumes, resulting in long computation times and large network sizes.
- **Noise:** When noise cannot be effectively distinguished from the underlying data, DL methods frequently have trouble in accurately predicting anomalies.

We will show how our solution, the **Anomaly Surfer**, is able to deal with these limitations while managing to perform just as well or even better than other state-of-the-art anomaly detectionmodels.

3. The Anomaly Surfer

3.1 Background

To best explain how our model works, we will first introduce an important concept from signal processing theory: the wavelet transform. To understand the wavelet transform, let us compare it to its more famous cousin, the Fourier transform. Consider the light signal emitted by a traffic light over the period of an hour. The Fourier transform would show us wavebands spanning the green, yellow, and red frequencies on the frequency spectrum. This is great if we want to know what signals were emitted, but it does not tell us exactly when the traffic light jumped to a certain color, only roughly how many times these different color signals occurred. To fix this, we could take smaller time intervals over which we compute the Fourier transform, but this could lead us to miss out on lower frequency wavebands spanning longer than the chosen interval. This is where the wavelet transform can help us out: in the traffic light situation, the wavelet transform not only shows us which wavebands roughly occurred, but also when these happened. More specifically put, the wavelet transform outputs a combination of time and frequency, which helps us in identifying the behavior of a signal at a certain time.

Our interest mainly goes out to the discrete version of the wavelet transform, where we can use it to function as an octave filter bank through the process of multi-resolutionanalysis. Here, our signal data can be passed through a series of high-pass and low-pass filters to generate a set of frequency sub-bands. Each filtering operation leads to a reduction in data size, effectively halving the frequency and creating an 'octave'. This creates a multi-scale representation of data, segregating high-frequency details from low-frequency components, which are called *high frequency* and *low-frequency coefficients*. In particular, these coefficients offer us many opportunities for further analysis, as each component can offer us insights into the behavior at different time-frequency levels.

Although the wavelet transform is clearly a promising method for time series analysis, there are some limitations that come with using the method as-is. One such limitation comes from the reliance on the choice of wavelet, which is a wave-like oscillation that is

localized in time and that is at the center of the high-pass and low-pass filter definitions. A wrong wavelet choice can lead to the transform not being able to capture all relevant information, which has great implications for any further processing and analysis of the results. To deal with this limitation, our solution features a deep learning network which has the ability to "learn" the right wavelet coefficients.

3.2 Our model

At the core of our model is the so-called sparse wavelet network. This network is instantiated such that it functions as a learnable cascade network, with functionality that is analogous to a complete octave filter bank. Here, wavelets constitute the kernels in convolutional neural networks (CNN), such that convolutions in the model are completely analogous to convolutions in the discrete wavelet transform. Since the wavelet transform can be inverted, we can use our decomposition result as an input to reconstruct the initial signal input. This notion is what enables us to train the model; we simply let it minimize the reconstruction loss between the signal output and the initial input. Moreover, by parameterizing an extra thresholding function on the high-frequency filters, we provide the option for the model to filter out noise. This all leads to a small but fast and flexible network that "learns" to decompose and reconstruct an input signal, such that as much information as possible is retained in the process.

After training the model on the signal input, we use a final network pass to retrieve the *high-* and *low-frequency coefficients*. These then feature as the inputs for the next step of the model. During this step, a window is slid over each coefficient, effectively creating a matrix consisting of a series of windows stacked on top of each other. Then, we fit a multivariate normal distribution by computing a sample mean and covariance over the matrix. Using this distribution, we can then determine the likelihood of each individual window occurring within the distribution. This likelihood is then compared as a percentage to a user-specified threshold; if the window is less likely to occur than the given threshold, it is deemed anomalous, resulting in a binary anomaly vector over all windows. To return to input dimensions, a roll-back is applied over this anomaly vector. After this process has been completed for all coefficient inputs, we make use of the fact that the decomposition result (i.e. the input coefficients) constitutes a full binary tree, meaning we can aggregate and cascade the anomaly score top-down through the decomposition levels to arrive at a final anomaly score vector of the same size as the input signal.

One limitation of the wavelet transform is its sensitivity to periodic data and time shifts. To negate this drawback, we introduce the concept of *similarity masking*. This optional method works only when we can specify a nonanomalous data interval as a reference for "good behavior" and uses the reconstructed signal as an input.

It starts by creating subsequences of a length chosen based on the most-frequent wavebands derived by using the Fast Fourier Transform. Then, it finds the most similar subsequence in the non-anomalous data interval using Mueen's Algorithm for Similarity Search (1). Based on this similarity, a normalized score is computed for each subsequence, such that subsequences that have high similarities receive a low score and are therefore considered non-anomalous. This mask is then applied on the anomaly score, leading to higher anomaly scores when subsequences do not share behavior with the non-anomalous data interval (which in reality often corresponds with the training data).

An overall depiction of how the model functions is given in Figure 3. Due to its sparse architecture and network, the model requires considerably less time and space compared to regular deep learning solutions. Furthermore, due to the second phase not involving any machine learning, parallelism is included with the model to bring down the computation time even further. This leads to a model that is extremely light-weight and fast, creating opportunities for it to be run online or even on edge devices. Moreover, the model can run both univariate as well as multivariate data inputs, where we can simply compute the score for each individual series in the multivariate set before we aggregate over all the scores to achieve a final anomaly score.

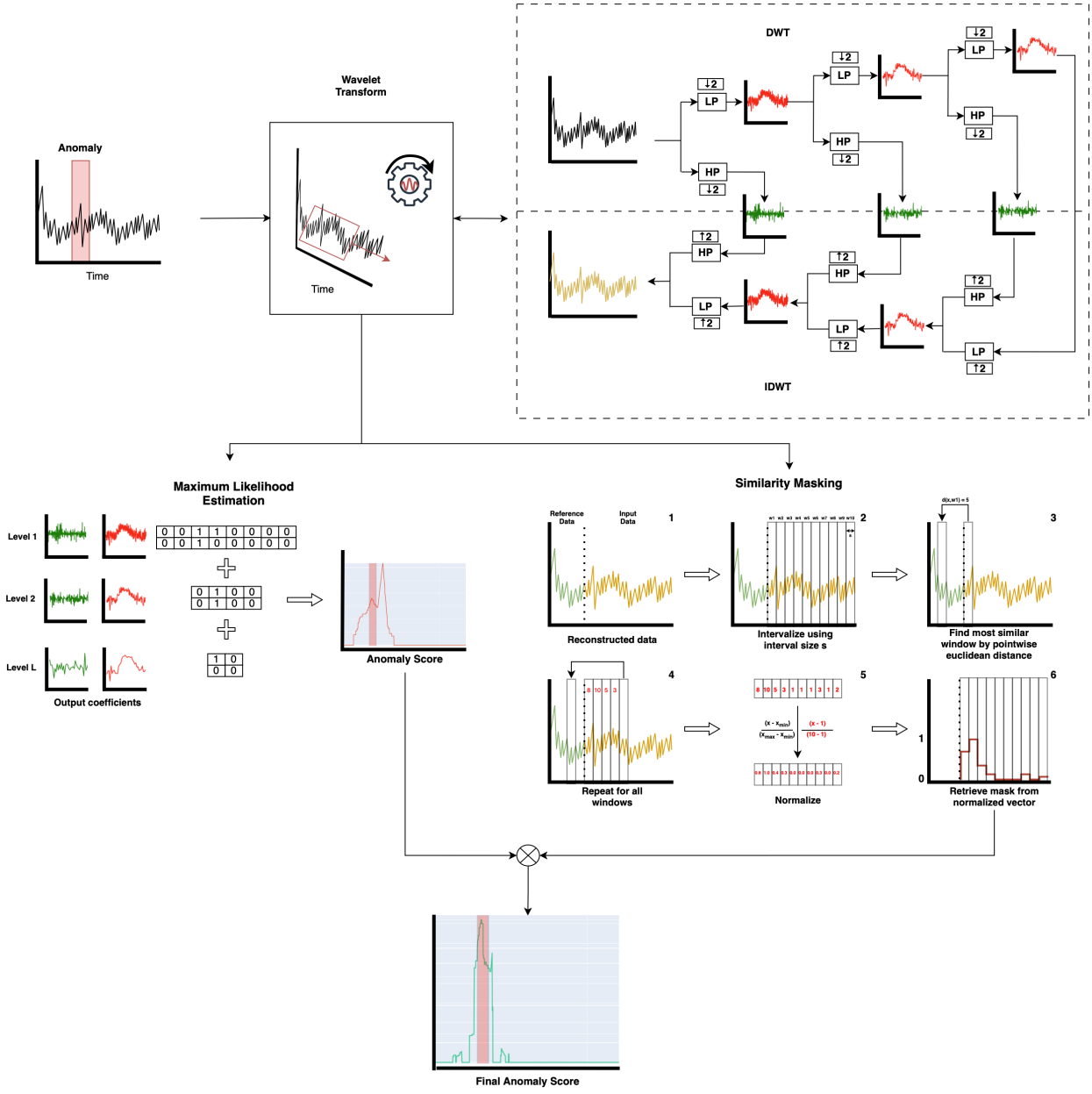


Figure 3: High-level overview of the Anomaly Surfer model.

4. Experiments and results

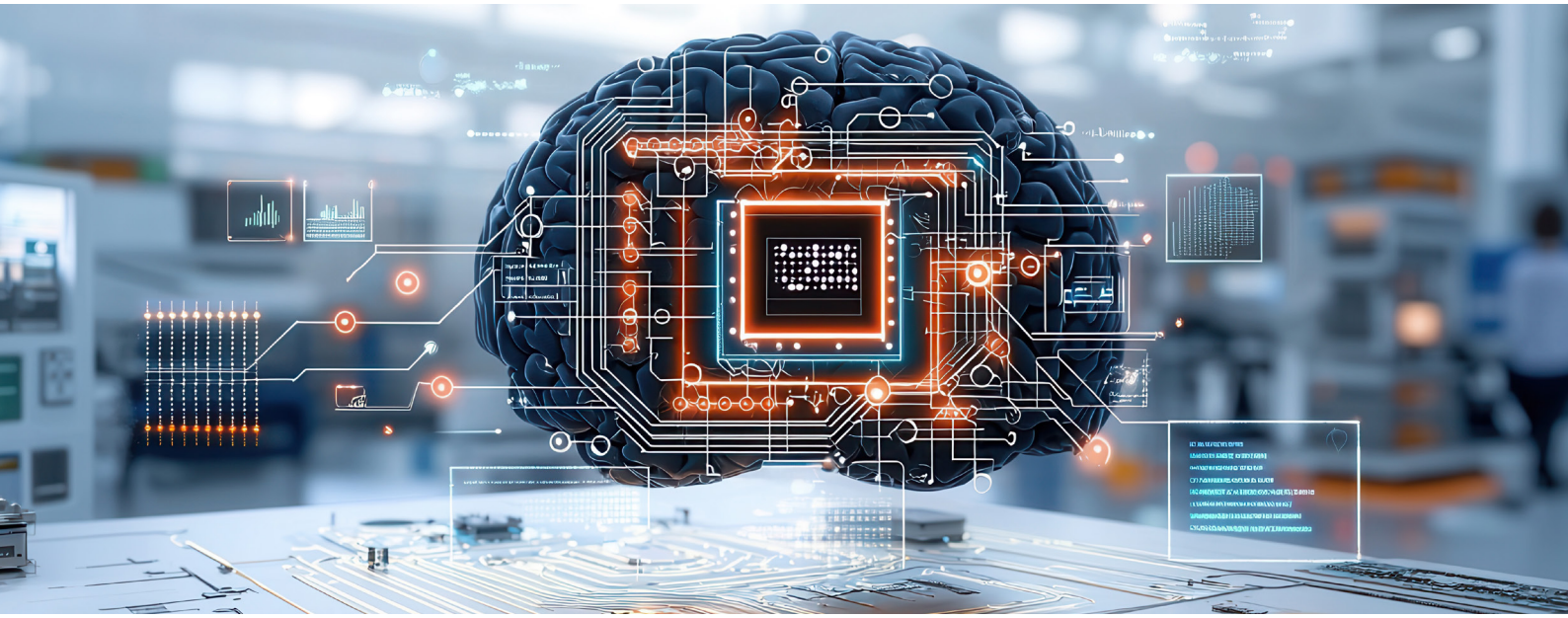
Evaluating anomaly detection models is not a trivial task. There exists a multitude of data archives that are often used in research to test the performance of anomaly detection models. There is even academic discourse over what the best evaluation metrics are. Since there is no golden rule to what data and what metrics are considered 'the best', we opt toinclude our model in a state-of-the-art benchmark suite (2) to provide a fair and accurate comparison to other models. This benchmark also includes two evaluation metrics that it suggests as being the most accurate (the "Volume Under Surface" / "VUS" metrics) and contains the largest compiled data archive for both univariate and multivariate data out there. In particular, it contains 770 individual univariate datasets and 200 individual multivariate datasets curated from 40 different data archives. Our experiment contains two different runs, one with a default parameter configuration and one where we naively fine-tune our noise reduction, likelihood threshold, and similarity masking parameters per individual dataset; by choosing the best run out of a few non-default configurations.

The results of running our model through the benchmark and the results of a few other selected models from research are shown in Table 1. The table clearly shows our model performing better than most state-of-the-art models on all evaluation metrics. Moreover, the results show that our model performs well even for default parameter settings. To show an example of how similarity masking can affect the anomaly score, Figure 4 displays the output of the model with and without similarity masking enabled.

	Method	AUC-PR	AUC-ROC	VUS-PR	VUS-ROC	Standard-F_1
Univariate	Sub-PCA (3)	0.37	0.71	0.42	0.76	0.42
	USAD (4)	0.32	0.66	0.36	0.71	0.37
	Sub-KNN (5)	0.27	0.76	0.35	0.79	0.34
	MatrixProfile (6)	0.26	0.73	0.35	0.76	0.33
	CNN (7)	0.33	0.71	0.34	0.79	0.38
	SR (8)	0.32	0.74	0.32	0.81	0.38
	OmniAnomaly (9)	0.27	0.65	0.29	0.72	0.31
	TimesNet (10)	0.18	0.61	0.26	0.72	0.24
	AutoEncoder (11)	0.19	0.63	0.26	0.69	0.25
	AnomalySurfer (default)	0.39	0.90	0.56	0.91	0.46
	AnomalySurfer (fine-tuned)	**0.47**	**0.91**	**0.64**	**0.92**	**0.53**
Multivariate	CNN (7)	0.32	0.73	0.31	0.76	0.37
	OmniAnomaly (9)	0.27	0.65	0.31	0.69	0.32
	PCA (3)	0.31	0.7	0.31	0.74	0.37
	LSTMAD (12)	0.31	0.7	0.31	0.74	0.36
	USAD (4)	0.26	0.64	0.3	0.68	0.31
	AutoEncoder (11)	0.3	0.67	0.3	0.69	0.34
	AnomalySurfer (default)	0.44	0.81	0.49	0.84	0.47
	AnomalySurfer (fine-tuned)	**0.48**	**0.87**	**0.54**	**0.90**	**0.51**

Table 1. Comparison of the mean evaluation results between the Anomaly Surfer and a selection of the best performers and well-known methods from research as provided by TSB-AD. (2)

Figure 4. Example output with and without similarity masking.



5. Implications and use-cases

The model's performance and flexibility make it suitable for use in a diverse set of use cases. The sparsity and relatively small computational requirements of the model make it applicable for both online and edge implementations. Due to its ability to discern behavior from noise, it can detect anomalies in very stochastic environments. Moreover, the model's generality implies it can be applied to other areas frequently related to anomaly detection, such as performance monitoring, fault detection in complex machinery, fraud detection, equipment failure prevention, and patient monitoring. For example, the model could be used to investigate the behavior of multiple data sources within complex machines. Since the model is context agnostic, it lends itself well to extensions and inclusions within other models. Therefore, if applied correctly, the model can be used within many different types of situations.

Despite these benefits and opportunities, the model is not a one-size-fits-all solution. More work remains on how to best use the model in real-time situations. Additionally, the model does require expert knowledge to configure correctly. Moreover, the value it creates is very dependent on the implementation and the context within which the model is used. For anomaly detection in general, domain knowledge is key to determine whether or not action should be taken when abnormal behavior is detected.

6. Conclusion

The Anomaly Surfer is a very promising anomaly detection model that has the potential to be used within many different scenarios and use cases. It has the ability to outperform most state-of-the-art time series anomaly detection methods in the literature and has relatively low time complexity when it comes to dealing with large data inputs. This creates opportunities for it to be applied in online and real-time settings. Moreover, the ability to filter out noise and deal with periodic data makes it a unique model that can deal with very stochastic environments and behavior. If you want to know more about the model or how it can be applied for your own use case, feel free to contact the Center of Excellence via centerofexcellence@ict.nl.

ACKNOWLEDGEMENTS

The Anomaly Surfer model was developed within the Center of Excellence.

Bibliography

1. Sheng Zhong and Abdullah Mueen. MASS: distance profile of a query over a time series.*Data Mining and Knowledge Discovery*, pages 1–27, 2024.

2. Qinghua Liu and John Paparrizos. The Elephant in the Room: Towards A Reliable Time-Series Anomaly Detection Benchmark. In *NeurIPS* 2024, 2024.

3. Charu C Aggarwal and Charu C Aggarwal. *An introduction to outlier analysis*. Springer, 2017.

4. 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*, pages 3395–3404, 2020.

5. Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 427–438, 2000.

6. Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In 2016 *IEEE 16th international conference on data mining (ICDM)*, pages 1317–1322. Ieee, 2016.

7. Mohsin Munir, Shoaib Ahmed Siddiqui, Andreas Dengel, and Sheraz Ahmed. DeepAnT: A deep learning approach for unsupervised anomaly detection in time series. *Ieee Access*, 7: 1991–2005, 2018.

8. Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-series anomaly detection service at Microsoft. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3009–3017, 2019.

9. 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*, pages 2828–2837, 2019.

10. Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

11. Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, pages 4–11, 2014.

12. Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, Puneet Agarwal, et al. Long short term memory networks for anomaly detection in time series. In *Esann*, volume 2015, page 89, 2015.

ICT High Tech
Center of Excellence
centerofexcellence@ict.nl
+31 (0)88 908 2000

ICT High Tech