Autoencoder and Kernel Density Estimation Based Approach for Time Series Anomaly Detection

Anton Arzha(✉) and Vladimir Korkhov

St. Petersburg State University, St. Petersburg 199034, Russia
st102714@student.spbu.ru, v.korkhov@spbu.ru

Abstract. The detection of anomalies in time series is a critical task across various domains including time series generated by a diversity of distributed systems. Despite extensive research, no single algorithm is the undisputed leader in all respects. This paper introduces KDEAE, a novel unsupervised approach for univariate time series anomaly detection. This method combines autoencoders (AE) and kernel density estimation (KDE). The proposed approach segments time series into non-overlapping segments, encodes them using convolutional AE and constructs anomaly scores based on their density obtained from KDE. The method is evaluated on GutenTAG dataset collection using TimeEval framework and compared with 25 algorithms of the same category. The results show that KDEAE outperform 40% algorithms by ROC AUC and 60% by PR AUC scores and 60% by runtime. The paper concludes with suggestions for further research to enhance the method's performance and extend it to multivariate time series.

Keywords: Anomaly Detection · Time Series · Autoencoders · Kernel Density Estimation

1 Introduction

The problem of identifying anomalies in time series data emerge in a vast number of applications, including industry, medicine, finance and others. In particular, infrastructures of distributed systems often continuously collect a large amount of information, including time-series information, which can be analyzed to optimize processes and respond to urgent conditions. The presence of anomalies often indicates various errors, unusual system operation mode and other non-standard situations, which makes this task so crucial. Detecting anomalies enables troubleshooting, risk mitigation, and promotes a deeper understanding of the target process.

Despite the fact that the problem of anomaly analysis has been studied for a long time, this field is actively developing, and to date there is no algorithm that could be called an absolute leader. Thus, the authors of [1] compared more than 70 methods and came to the conclusion that a significant part of them has problems related to reliability and complexity of tuning a large set of hyperparameters.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15894, pp. 249–263, 2026.
https://doi.org/10.1007/978-3-031-97648-3_17

Manual annotation of real time series data is often an almost impossible task due to the huge amount of data or because true labels are simply unknown [2]. Therefore, anomaly detection approaches related to the unsupervised learning paradigm are of great interest because of their ability to discover natural patterns in the data without requiring training samples**.**

A substantial part of the models takes as input a time series in the form of subsequences of equal length, thus applying a sliding window approach. This approach is justified because in real data anomalies are usually not located at a single point, but are a longer-term pattern [3].

Neural networks of different architectures are widely used to discover anomalies in time series [4]. One of such architectures are autoencoders – neural networks that can perform non-linear compression of data. The most obvious AE-based approach for detection of anomalies is to use reconstruction error as a measure of anomalousness. However, recent studies criticize this method because of the unwanted reconstruction of anomalies that autoencoders can exhibit [5]. This problem can be mitigated by training the model solely on normal data, but this imposes additional limitations on the applicability of the methods.

Alternatively, hidden representations can be used directly to detect anomalies. For example, kernel density estimation applied to the data compressed by AE provides its distribution density, which can be used to detect anomalies. Using this approach is feasible because the anomalies are assumed to be rare.

In this paper, we propose a method for univariate time series anomaly detection based on AE and KDE, named KDEAE. The essence of this approach is to divide time series into segments of equal length and training AE to encode them. The values of empirical probability density function derived from KDE are used to construct anomaly scores of each subsequence of the time series.

The evaluation of our algorithm was conducted using TimeEval tool [6] which makes it easy to test and benchmark the anomaly detection methods on multiple datasets. The GutenTAG dataset collection from the same researchers, which contains many synthetic time series with anomalies of various types, was used as test data.

The source code and evaluation results are available in repository [7].

The rest of the paper is organized as follows. In Sect. 2, we summarize similar approaches and other existing solutions that address the same problem. In Sect. 3 we provide a comprehensive explanation of the proposed algorithm including data preparation, model architecture, training process and gaining anomaly scores from density obtained by KDE on autoencoder hidden representations. In Sect. 4 we outline and discuss the results of evaluation of our model and compare it with other algorithms. In Sect. 5, we consider the performance of KDEAE on several sample time series. Finally, in Sect. 6 we conclude the paper and suggest directions for further research.

2 Related Work

A number of AE-based models have been developed for time series anomaly detection. In [8] the authors utilized long short-term memory (LSTM) autoencoder for discovering anomalies in multivariate real industry data and achieved high quality predictions. In

addition, it is important to note the significance of the variational autoencoder (VAE). Such models are trained to describe latent space in probabilistic terms. Anomaly discovery method based on LSTM-VAE was introduced in [9] and showed outstanding results for sensor data.

Combining AE and KDE for anomaly detection in tabular data has been successfully implemented in [10]. The authors trained AE on normal data, estimated its density with KDE and calculated threshold for anomaly detection. The approach was tested on several datasets and demonstrated high ROC AUC scores. For time series data, a novel approach was proposed in [11] with the key idea to apply KDE on the reconstruction errors of the AE. The obtained density is then used to gain anomaly scores. Experiments showed that using KDE instead of just reconstruction error significantly increased performance by nearly 10%.

In general, there are numerous unsupervised methods for time series anomaly detection based on different approaches. For example, there are algorithms for computing a matrix profile, such as STAMP [12] and its modifications. Their idea is to find the nearest neighbor by distance for each subsequence of the time series. Some distance-based approaches build a model of normal behaviour and compare subsequence to it to detect anomalies. For example, SAND [13], designed to process streaming data, incrementally update such model while also being able to compute anomaly scores at every step. Forecasting methods are trained to predict the value of a time series at the next point(s) and then compare the expected values with the actual values. Most of these methods require a normal time series for training, but there are also models like ARIMA [14] that can operate in unsupervised mode. Some methods successfully implement rarely used approaches. For example, Series2Graph [15] builds a graph from subsequence embeddings, with which it detects anomalies as they form less frequent patterns. DWT-MLEAD [16], by contrast, applies wavelet transform to data and discover anomalies by log-likelihood.

Overall, how diverse and rapidly evolving the field of time series anomaly detection is emphasizes the relevance of this paper as well.

3 Methodology

3.1 Preliminaries

An autoencoder is a neural network consisting of two sub-networks, often having a symmetric structure. The first of them, the encoder, reduces the dimensionality of the input data, i.e., encodes it. The second, the decoder, reconstructs the input data from the encoded representation. Thus, the AE is trained to perform nonlinear data compression in unsupervised mode.

The convolutional AE is based on convolutional layers. In particular, when analyzing sequences, one-dimensional convolutions in the encoder and one-dimensional transposed convolutions in the decoder are used.

KDE is a non-parametric method to estimate the probability density function of given independent and identically distributed samples. For $x_1, x_2, \ldots, x_n, x_i \in \mathbb{R}^k$ their

density estimation $\hat{p}(x)$ using Gaussian kernel $K(u)$ and bandwidth h is given as in the following formula:

$$\hat{p}(x) = \frac{1}{N}\sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right), K(u) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}u^2\right)$$

3.2 Model Architecture

AEs can typically downsize time series dimensionality by a factor of several. The latent space should have enough features to separate normal and anomalous subsequences but must not be too large not to suffer from curse of dimensionality. Aiming to create some kind of fixed architecture model we choose model latent space to consist of 4 dimensions, which satisfies the requirements above. Therefore, to be effectively encoded, the input sequences must have length about couple of dozens. We choose this parameter to be equal exactly 23, to avoid zero padding in decoder's transposed convolution layers.

The resultant model encoder consists of three convolutional layers with a ReLU activation between them and a linear layer. The decoder mirrors the same structure but use transposed convolutional layers. To reduce data dimensionality at each step, the stride of convolutional layers is chosen to 2. The architecture of model layers with tensor shapes after each block is displayed in Fig. 1.

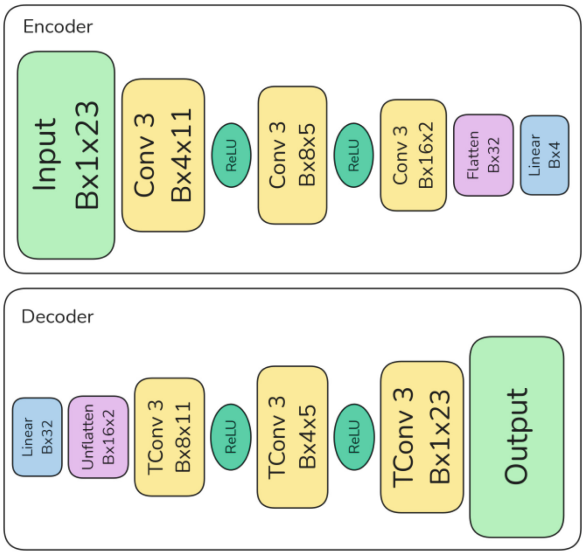


Fig. 1. Architecture of the proposed model

The most authors utilizing sliding window approach to analyze time series work with intersecting windows (most often, with stride 1) thus increasing model resolution.

Unlike them, we work with non-intersecting segments of time series. In the following sections we show that this approach can perform well while also significantly reducing the computational complexity.

Another important aspect is that we assume training the model on all available data, not just normal data, which is often the case in other works. This is done so that it is possible to apply the algorithm without having any additional knowledge about the available data. To find the optimal number of training epochs, the data can be divided into training and test samples and the number of epochs at which overfitting is not significant can be chosen. Overfitting deteriorates the quality of predictions, because latent features in this case begin to encode not only the desired data structure, but also noise.

3.3 Building Anomaly Score

The process of obtaining anomaly scores from AE's encoded representations consists of several steps, which will be explained in detail below.

1. Z-normalize encoded representations and find *middle* point
2. Calculate KDE bandwidth using quantile q of distances to the middle point
3. Apply KDE and perform robust z-normalization of its results
4. Apply sigmoid* and expand scores to the original length of time series

The important problem of the KDE-based approaches is finding kernel bandwidth. Since our goal is to separate anomalies, we should choose not bandwidth value that gives the most likely probability density function, but the one that improves the performance of the method. Our idea is to control kernel width using distances from some specific point to others. That specific point, called middle point, is the closest data point to medians of all dimensions. The bandwidth is chosen as quantile q (also referred as bandwidth parameter) of distances to the middle point. If KDE kernel has limited width, that parameter would be equal to the fraction of points which will affect density value of middle point, but since we use Gaussian kernel, the interpretation is vaguer.

After achieving KDE scores, we could just use min-max normalization to get anomaly scores, but such approach will lead to a large difference of scores for various datasets, at which anomalies would be found. Moreover, min-max normalization will always rank some points with 100% (and 0%) level of anomaly even if they are not so in fact. To try to bring some standardization to these values, robust z-normalization is firstly applied to the KDE output values and then sigmoid (*with growth parameter equal to -1, so that rarer samples have a higher anomaly rate) is providing final scores for subsequences. The last step is expanding window scores to the original length of time series, taking point anomaly level equal to level of the corresponding segment.

The obvious disadvantage of this approach is that despite all the transformations, essentially a ranking of points by anomalousness is performed. Thus, even if there are no anomalies in the time series under consideration, this algorithm is likely to identify some points with a high level of anomalies.

4 Model Evaluation

To rate mode efficiency, we will test in on the GutenTAG dataset collection using TimeEval framework [6]. This dataset contains 168 univariate time series (each have a length of 10,000 points) with anomalies of various types to comprehensively evaluate the quality of the model in different cases. Moreover, in [1], the authors tested many algorithms in the same way, so that it will be possible to evaluate the proposed model in comparison with other existing approaches, we will use their results for other algorithms. The model will be evaluated using ROC-AUC, PR-AUC and runtime. We will consider how these metrics depends on the number of *epochs* to train autoencoder and quantile q which is used to determine KDE bandwidth.

4.1 Parameters Selection

As for the number of epochs, for a single time series it is better to select this value by dividing the data into training and test samples. This is due to the fact that for some simple dependencies, several dozens of epochs are enough for the model, while in some cases several hundred epochs are not sufficient. Nevertheless, in order to obtain average estimates for different time series, we trained the model with different but fixed for all time series number of epochs.

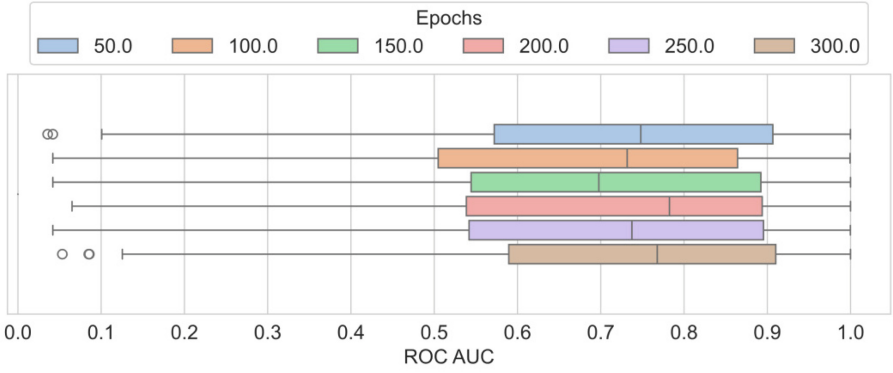


Fig. 2. Model ROC AUC vs Epochs

Figures 2 and 3 shows how ROC AUC and PR AUC respectively are changing when variating epochs parameter, (bandwidth parameter is chosen to be 0.004). Despite values are close to each other, it is clearly seen that 200 epochs provide better scores so we consider this value as standard for that dataset collection.

Another important aspect is model runtime which is controlled mostly by the number of epochs. For 200 epochs, the average runtime is 1.77 ms per data point and it is linearly changing with the number of epochs since it is the most computationally complex stage of the algorithm.

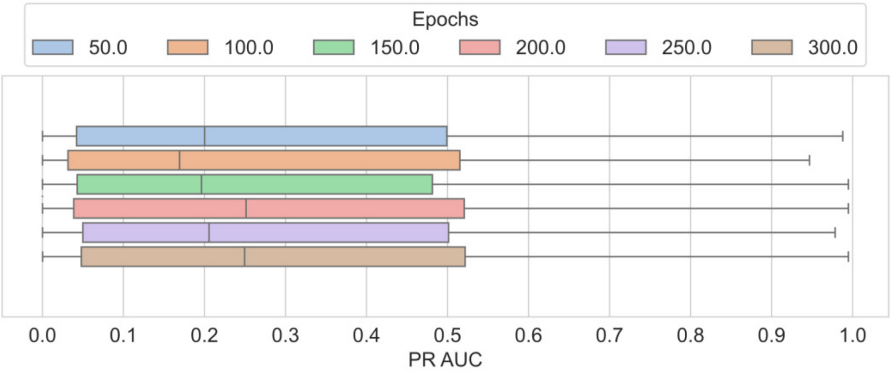


Fig. 3. Model PR AUC vs Epochs

The most important model parameter is bandwidth parameter q. Experiments show that the best value of this parameter in terms of ROC AUC is 0.001, or about the same order of magnitude. If this parameter is increased too much, the model quickly becomes a random guesser. Similarly, too small values decrease efficiency as well. Since the model was tested on time series of the same length from one collection, further experiments are required to identify patterns for proper parameter selection in different cases. Figures 4 and 5 shows how target metrics depends on that bandwidth parameter. Below, unless otherwise noted, the value is selected to be 0.001. It can be also seen that PR AUC is even better with smaller values, but since ROC AUC in that case significantly deteriorates, we do not consider that values.

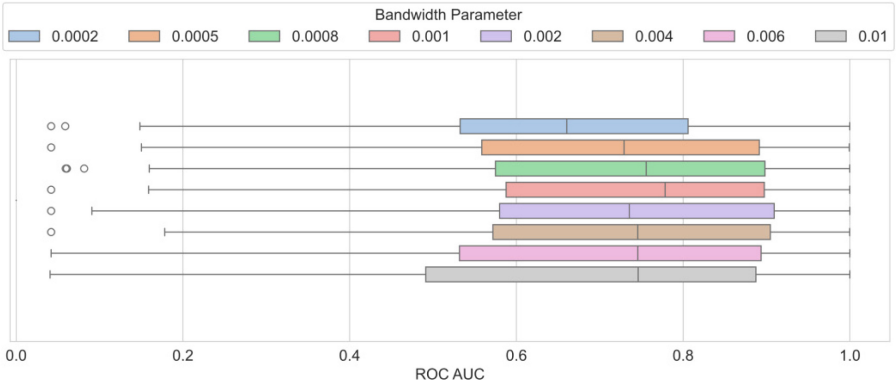


Fig. 4. Model ROC AUC vs Bandwidth Parameter

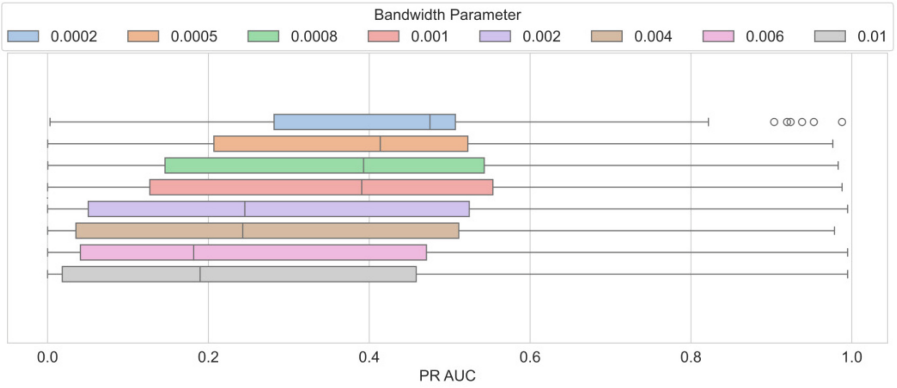


Fig. 5. Model PR AUC vs Bandwidth Parameter

4.2 Comparison with Other Algorithms

The following figures compares our method KDEAE with other 25 unsupervised univariate anomaly detection methods evaluated in [1] on the same data in terms of ROC AUC, PR AUC and runtime.

Figures 6 and 7 demonstrates that KDEAE outperformed 40% of algorithms by ROC AUC and 60% by PR AUC score. These results are promising, and presumably can be improved with further research.

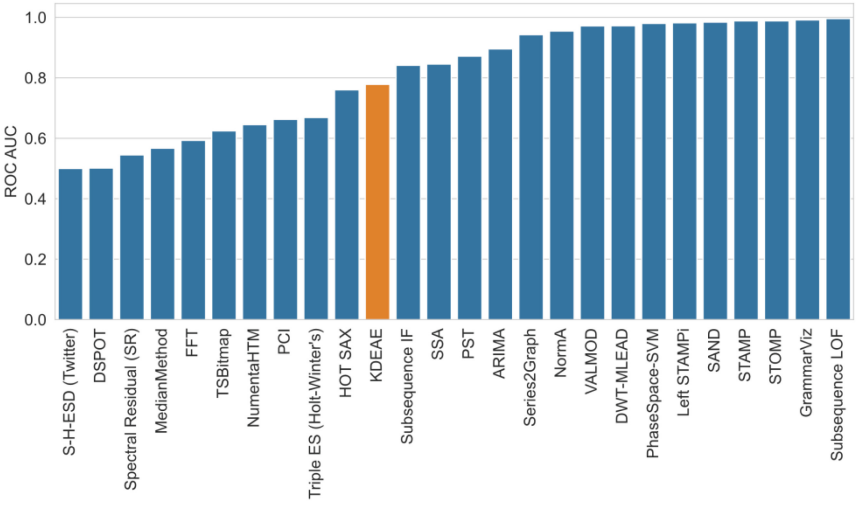


Fig. 6. Median ROC AUC Comparison

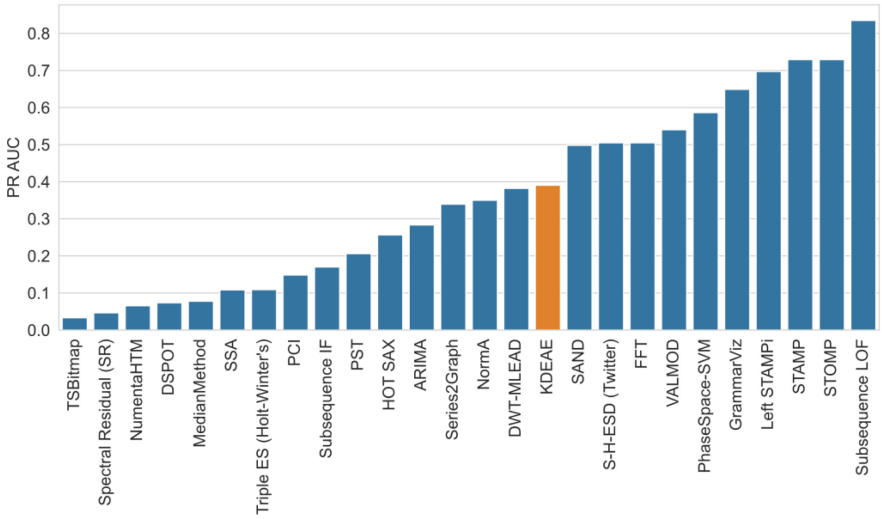


Fig. 7. Median PR AUC Comparison

KDEAE also outperform 60% of algorithms by runtime, which can be seen in Fig. 8. It is also notable that only one algorithm (STOMP) performs better that ours on all three metrics. In most cases, there can be seen a tradeoff runtime-quality.

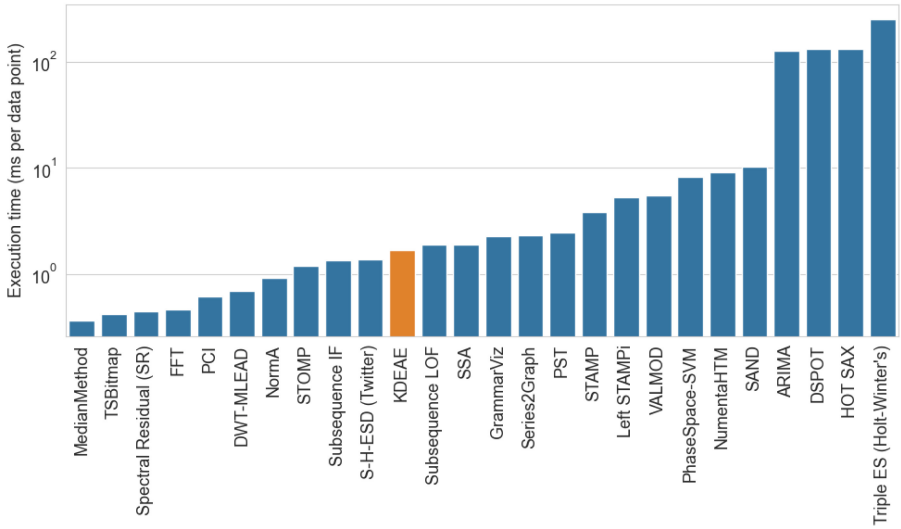


Fig. 8. Runtime Comparison

4.3 Efficiency for Different Types of Anomalies and Base Oscillations

In this section, we evaluate the performance of KDEAE for different types of anomalies and base oscillations, following dataset authors' [6] classification. For base oscillations we use abbreviations: poly – polynomial, rw – random walk, cbf – cylinder-bell-funnel, ecg – electrocardiogram.

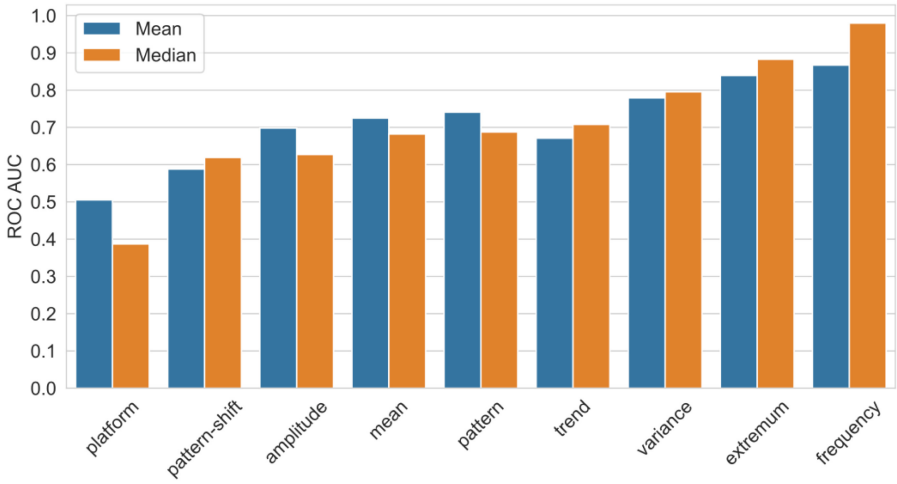


Fig. 9. ROC AUC vs Anomaly Type

Figure 9 shows that KDEAE performs best in terms of ROC AUC for frequency, extremum and variance anomalies. However, it is not suitable for platform anomalies.

Similar pattern can be observed in Fig. 10, but relative to PR AUC. The highest scores are again on frequency anomalies. We see that KDEAE is the least effective to discover trend and platform types.

As for base oscillations, the ROC AUC scores don't vary that much, which can be seen in Fig. 11. Nevertheless, they are generally better for periodic functions. On the other hand, the PR AUC values take a larger range. Figure 12 shows that again the best scores are for periodic series. However, polynomial data have significantly lower metric values in that case.

Overall, experiments demonstrate that KDEAE is suitable to detect anomalies of different types and base oscillations. However, both metrics show low efficiency for platform-type anomalies. It is also notable, that the results are generally better for periodic time series, which is a property of many such algorithms.

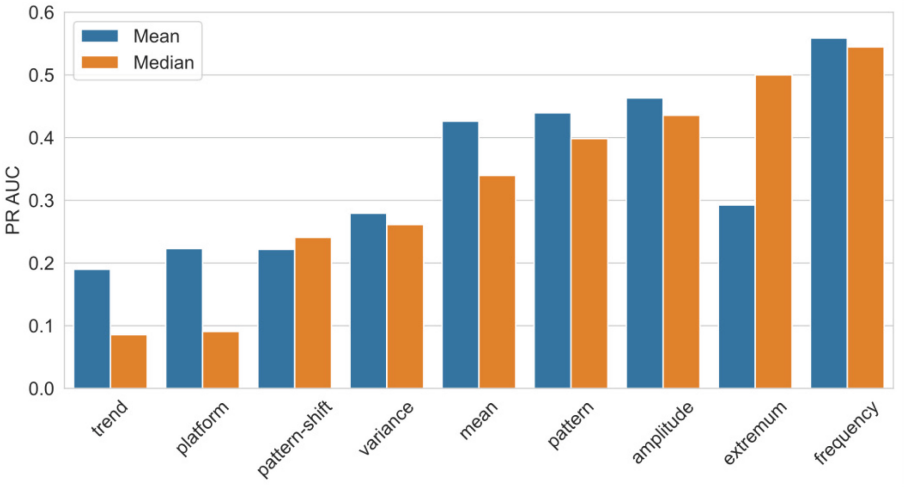


Fig. 10. PR AUC vs Anomaly Type

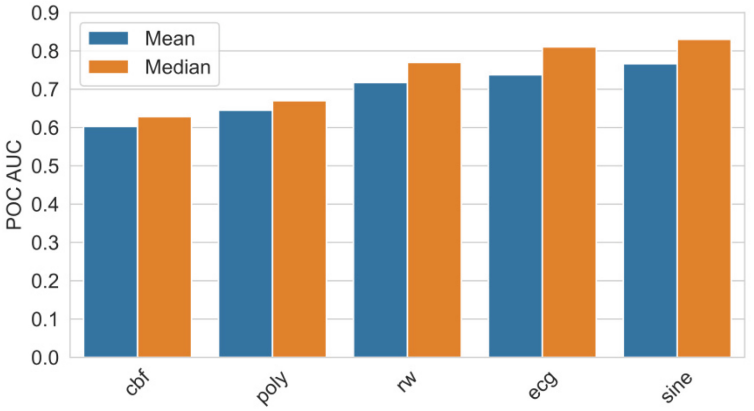


Fig. 11. ROC AUC vs Base Oscillation

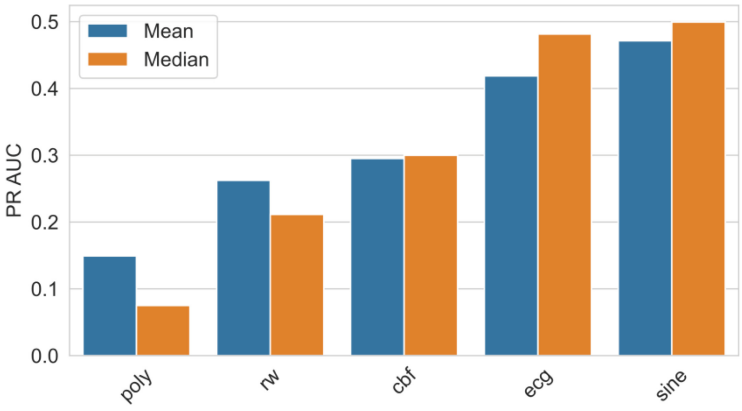


Fig. 12. PR AUC vs Base Oscillation

5 Examples of Model Performance

This section provides examples of anomaly detection using KDEAE. Since time series are too large to display in their entirety, a small section of time series containing anomaly is shown. The anomaly threshold is determined using a specific quantile of the obtained estimates. In general, the larger the threshold quantile is, the better the method performs. Thus, at $q = 0.99$, true anomalies obtained the highest anomaly scores. As the quantile decreases, the number of false positives increases. The specific quantile used for each plot is shown in its legend. The bandwidth parameter was adjusted to improve the performance. Since the algorithm treats the input data as fixed windows, small shifts between true and detected anomalies are possible.

As was stated before, performance is generally better for periodic time series.

Figure 13 shows example of outstanding KDEAE performance for ECG-type time series with frequency anomaly.

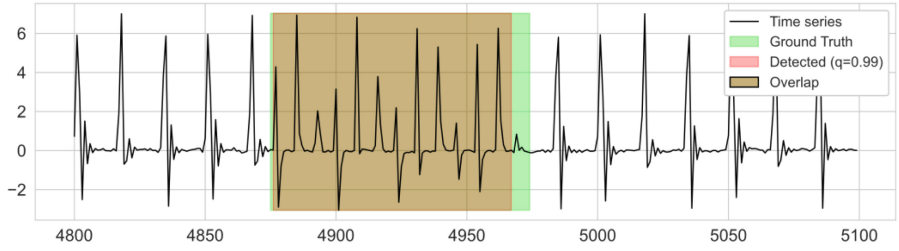


Fig. 13. Sample frequency anomaly detection for ECG type signal

Example of detection pattern anomaly for sine-based time series is depicted in Fig. 14. For that sample true anomaly also obtained the highest anomaly score.

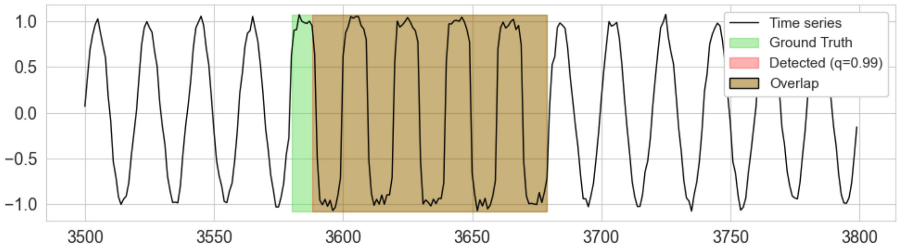


Fig. 14. Sample pattern anomaly detection for sine type signal

For non-periodic series, anomalies can also be discovered, but often with lower thresholds resulting in false positives. For example, Fig. 15 shows example how KDEAE discover anomaly in cbf-based time series.

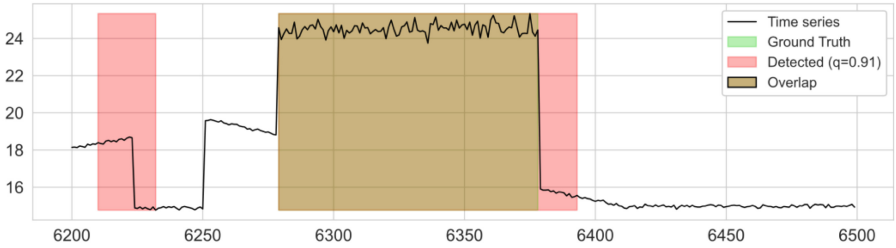


Fig. 15. Sample anomaly detection for cbf-based time series

Figure 16 shows an example of anomaly detection for random-walk-based time series. With high threshold, anomaly was detected only partially.

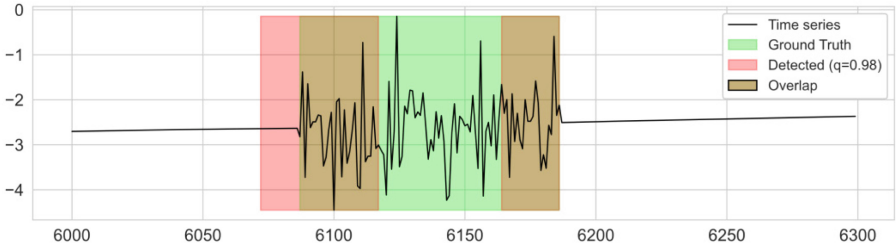


Fig. 16. Sample anomaly detection for random-walk-based time series

The weakness of the algorithm is the detection of platform type anomalies. Since windows related to a platform are very similar to each other, they get similar encodings and hence a higher density estimated by KDE. In this case, the basic assumption of low anomaly density does not work. Anomalies of this type can only be detected if there is

a stronger pattern in the time series, such as periodicity or stationarity. Otherwise, their detection is unlikely. An example of this behavior of the algorithm is shown in Fig. 17.

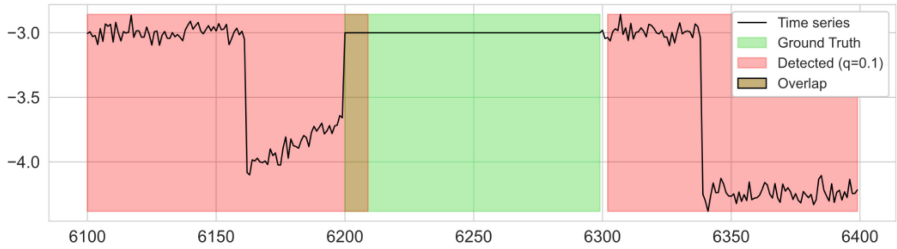


Fig. 17. Sample of platform-type anomaly detection

Overall, KDEAE can be used to discover various anomalies in both periodic and non-periodic time series. The disadvantage of the algorithm is the difficulty in detecting platform-type anomalies.

6 Conclusion

In this paper, we presented KDEAE, a novel approach for unsupervised time series anomaly detection that combines the strengths of autoencoders and kernel density estimation. The methods segments time series into non-overlapping subsequences, utilizing sliding window approach. These parts are then encoded by convolutional autoencoder. Anomaly scores are constructed based on the density of the encoded representations.

We evaluated our method on GutenTAG [6] dataset collection and compared in with 25 algorithms from the same category [1]. The results show that KDEAE is outperforming 40% of them by ROC AUC score, 60% by PR AUC score, and 60% by runtime. The results of the study demonstrate that combining AE and KDE is a promising approach for time series anomaly detection. The directions for further research could be optimizing the process of choosing parameters for the improvement of performance and extending the method to multivariate time series.

Acknowledgement. Supported by Saint Petersburg State University, project ID: 95438429.

References

1. Schmidl, S., Wenig, P., Papenbrock, T.: Anomaly detection in time series: a comprehensive evaluation. PVLDB **15**(9), 1779–1797 (2022). https://doi.org/10.14778/3538598.3538602
2. Belay, M.A., Blakseth, S.S., Rasheed, A., Salvo Rossi, P.: Unsupervised anomaly detection for IoT-based multivariate time series: existing solutions, performance analysis and future directions. Sensors **23**(5), 2844 (2023). https://doi.org/10.3390/s23052844
3. Wang, F., Jiang, Y., Zhang, R., Wei, A., Xie, J., Pang, X.: A survey of deep anomaly detection in multivariate time series: taxonomy, applications, and directions. Sensors **25**(1), 190 (2025). https://doi.org/10.3390/s25010190

4. Choi, K., Yi, J., Park, C., Yoon, S.: Deep learning for anomaly detection in time-series data: review, analysis, and guidelines. IEEE Access **9**, 120043–120065 (2021). https://doi.org/10.1109/ACCESS.2021.3107975
5. Bouman, R., Heskes, T.: Autoencoders for Anomaly Detection are Unreliable. arXiv preprint arXiv: 2501.13864 (2025)
6. Wenig, P., Schmidl, S., Papenbrock, T.: TimeEval: a benchmarking toolkit for time series anomaly detection algorithms. PVLDB **15**(12), 3678–3681 (2022). https://doi.org/10.14778/3554821.3554873
7. Project repository. https://github.com/UnicornTowa/KDEAE. Accessed 28 Mar 2025
8. Hsieh., R.-J., Chou, J., Ho, C.-H.: Unsupervised online anomaly detection on multivariate sensing time series data for smart manufacturing. In: 2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA), pp. 90–97. IEEE, Kaohsiung (2019). https://doi.org/10.1109/SOCA.2019.00021
9. Park, D., Hoshi, Y., Kemp, C.C.: A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. IEEE Robot. Automation Lett. **3**(3), 1544–1551 (2018). https://doi.org/10.1109/LRA.2018.2801475
10. Cao, V.L., Nicolau, M., McDermott, J.: A hybrid autoencoder and density estimation model for anomaly detection. In: Handl, J., Hart, E., Lewis, P., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) Parallel Problem Solving from Nature – PPSN XIV. PPSN 2016. Lecture Notes in Computer Science(), vol. 9921. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45823-6_67
11. Frehner, R., Wu, K., Sim, A., Kim, J., Stockinger, K.: Detecting anomalies in time series using kernel density approaches. IEEE Access **12**, 33420–33439 (2024). https://doi.org/10.1109/ACCESS.2024.3371891
12. Yeh, C.-C.M. et al.: 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), Barcelona, Spain, pp. 1317–1322 (2016). https://doi.org/10.1109/ICDM.2016.0179
13. Boniol, P., Paparrizos, J., Palpanas T., Franklin, M.J.: SAND: streaming subsequence anomaly detection. Proc. VLDB Endow. **14**(10), 1717–1729 (2021). https://doi.org/10.14778/3467861.3467863
14. Hyndman, R.J., Athanasopoulos, G.: Forecasting: Principles and Practice, (2nd edn.) OTexts (2018)
15. Boniol, P., Palpanas, T.: Series2Graph: graph-based subsequence anomaly detection for time series. Proc. VLDB Endow. **13**(12), 1821–1834 (2020). https://doi.org/10.14778/3407790.3407792
16. Thill, M., Konen, W., Bäck, T.: Time series anomaly detection with discrete wavelet transforms and maximum likelihood estimation. In: Proceedings of the International Conference on Time Series (ITISE) (2019)