technische universität dortmund

fakultät für informatik

Master Thesis

DEAN-TS: Deep Ensemble Anomaly Detection for Time Series

Tim Katzke

10.07.2023

Academic Advisors:
Prof. Dr. Emmanuel Müller
Simon Klüttermann, M.Sc.

Technical University of Dortmund
Department of Computer Science
Chair of Data Science and Data Engineering
https://ls9-www.cs.tu-dortmund.de/

Contents

List of Figures

List of Tables

1 Introduction

In today's data-driven world, the ability to detect anomalies in time series data has become a critical task across various domains such as finance, cybersecurity, industrial monitoring, and healthcare [34, 45, 49, 36, 40, 15]. Anomalies, sometimes defined as instances that deviate significantly from the expected behavior [18], may carry valuable insights and hidden patterns that can lead to improved decision-making, preventive measures, and enhanced operational efficiency [1]. Consequently, there is an increasing demand for accurate and efficient techniques to identify anomalies in complex and dynamic time series datasets.

While there exists dedicated foundational literature [1] and a plethora of novel anomaly detection methods are regularly being proposed, recent independent survey evaluations suggest that universally well performing, robust anomaly detection for time series data is still an open problem [43, 10].

Traditional methods for anomaly detection in time series data often rely on statistical techniques, manual feature engineering and domain expertise. While these approaches have provided valuable insights they can struggle to capture the intricate patterns and nuanced relationships present in more complex datasets [33]. As a result, there is a growing need for more advanced techniques that can adapt to the evolving nature of time series data and overcome the limitations of traditional methods.

Subsequently, deep learning approaches for time series anomaly detection have gained significant attention and shown promising results in various domains [21, 45, 15]. However, a critical assessment reveals several limitations and challenges associated with these methods as well [43]. Besides the commonly higher data and computational requirements when using these methods, interpreting the final models can often prove difficult.

Motivated by the potential of deep learning and the promises of ensemble methods to offset some of its shortcomings, this thesis aims to adapt the novel *DEAN* anomaly detection method, first devised for image data, to the time series domain [25]. The main idea is to retain the underlying concept of DEAN, namely to utilize an diverse ensemble of comparatively weak but deep neural networks utilizing a novel loss function paired with distinct restrictions. By combining the representational power of a deep learning architecture with the robustness of ensemble methods, we seek to devise an accurate, scalable and interpretable time series anomaly detection method.

1.1 Research Question and Contributions

The primary research question to be answered in this thesis is whether or not the underlying principles of DEAN can be successfully transferred to the time series domain. For this

purpose, we develop an adapted methodology that takes into account the peculiarities of temporal dependencies in the data. Besides the general adaptation of the basic approach for anomaly detection, we also consider the incorporation of time series decomposition as an additional preprocessing step to improve both performance and interpretability of the predictions.

Through extensive experiments on various synthetic and real-world datasets, we evaluate the proposed methodology in several variations and compare it with state-of-the-art methods. The obtained results demonstrate the effectiveness of our novel approach, revealing its ability to accurately and efficiently detect different types of anomalies, as well as certain limitations. Additional experiments explore relevant general properties of the adapted methodology and provide an assessment of the potential for further research.

1.2 Structure of this Thesis

In the beginning, Chapter 2 provides an introduction to the fundamental principles of time series anomaly detection, commonly used methods in this domain as well as performance evaluation metrics, laying the groundwork for subsequent chapters.

With Chapter 3, we delve into the original DEAN method. We examine the general DEAN methodology, its novel interpretability approach leveraging the underlying ensemble structure and summarize the empirical insights gained in the original work.

Building upon the original method, Chapter 4 outlines the strategies employed to adapt DEAN for time series anomaly detection. After first formulating the objectives and challenges of this endeavor, the modified DEAN-TS approach is described, including the incorporation of time series decomposition and its applicability for prediction interpretability.

To provide a comprehensive evaluation of the anomaly detection performance and runtime efficiency of DEAN-TS, Chapters 5 and 6 present benchmark results on diverse real and synthetic datasets in a semi-supervised and unsupervised setting. A comparative analysis is conducted, assessing the performance of different DEAN-TS versions against state-of-the-art anomaly detection methods and the impact of anomalous training data.

In Chapter 7, we conduct further experiments to analyze various properties of the developed methodology. These include the evaluation of individual submodels and the overall ensemble structure, as well as investigating the impact of the chosen combination method for combining the submodels. We also look at the potential benefits of incorporating preprocessing through time series decomposition.

Chapter 8 concludes the thesis by summarizing the results obtained throughout the study. Additionally, an outlook on potential avenues for future research questions is provided, highlighting areas that warrant further exploration.

2 Time Series Anomaly Detection

This chapter introduces the fundamentals of time series anomaly detection that are essential for the remainder of this thesis. A brief introduction to time series definitions as well as common deep learning and ensemble approaches in this domain outlines some of the methodological concepts taken up in the Chapters 3 and 4. The overview of common method families for time series anomaly detection and the performance metrics to quantify success in detecting anomalies provide necessary background for the experiments in the Chapters 5, 6 and 7.

2.1 Foundations of Time Series

A *univariate time series (UTS)* is a time-ordered sequence of observations, each corresponding to a single variable measured at different points in time [10]. This notion can be formalized as $X = (X_1, X_2, ...X_t)$ were X_i denotes the observed variables value at the time i for the time series X. For a *multivariate time series (MTS)*, each observation corresponds to multiple, simultaneously measured variables. Accordingly, the above definition can be generalized to $X_i = (X_{i,1}, X_{i,2}, ..., X_{i,n})$, where each X_i is a data point represented by a vector of n observed variables at time i.

The time delay or shift between observations in time series is sometimes called *lag* [5]. This value represents the number of time periods by which a data point is shifted. A lag k would therefore correspond to X_{i-k} for a given X_i, meaning in particular a positive lag value indicates comparison to a previous data point. Examining the relationship between observations at different lag values is commonly done to uncover dependencies, trends, and patterns within the time series data.

Instead of sequentially analyzing singular values or binary relations for a given lag, *window-based* techniques aims to analyze potentially longer periods of consecutive data points at once. For this, a window of fixed or varying size is slid in usually equidistant steps through the time series to derive relevant information for the currently observed data subset. This allows for a more direct examination of the higher level structural dynamics of the data.

If the overall behavior and statistical characteristics of the time series are time-invariant it is called *stationary*. This means the underlying statistical properties of the data are not dependent on the time of observation, or more formally, its unconditional joint probability distribution does not change.

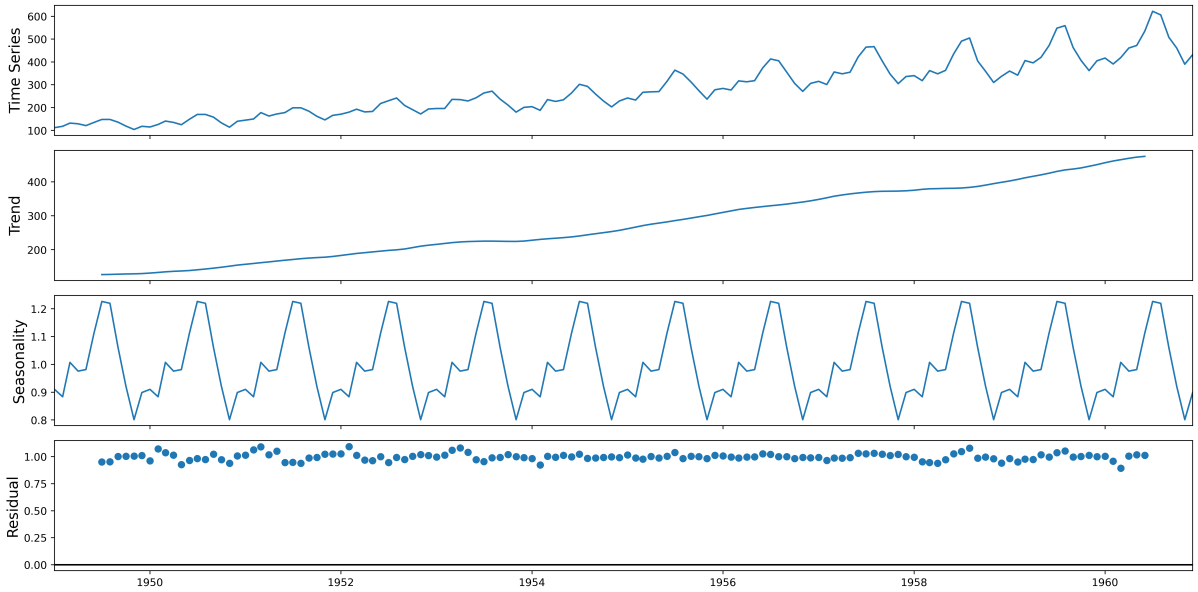


Figure 2.1: Representation of a univariate time series displaying a steadily increasing total number of airline passengers [13] next to the result of its decomposition into a trend, seasonality and residual component.

2.1.1 Time Series Decomposition

Structural patterns of a time series can be semantically decomposed into several components, each encapsulating different behavioral properties of the data [22]. These components are presented below and illustrated in Figure 2.1, using as example the growing number of international air passengers between 1949 and 1960.

The *trend* of a time series represents the long-term pattern or directionality exhibited by the data over time. It represents the underlying, gradual change in the series, like the overall steadily increasing number of airline passengers, that may occur due to various factors such as economic growth, population trends, technological advances or other influences. A trend can be ascending (increasing), descending (decreasing), or exhibit no clear direction (horizontal) and may incorporate a *cyclical* component, which reflects repeated but non-periodic fluctuations.

Seasonality refers to a repetitive and predictable pattern that occurs at fixed intervals within a time series. These patterns may be influenced by factors such as weather, holidays, or recurring events. Seasonality can manifest as regular fluctuations, cycles, or periodic variations in the data. For example, the number of airline passengers tends to increase during the summer season each year. Understanding seasonality is crucial for capturing and modeling these recurrent patterns accurately [22].

Residuals, also known as remainders, errors or noise, are the random fluctuations or variability in a time series that cannot be explained by the remaining components. They

reflect differences between the observed values and what would be expected by combining the trend and seasonality components. Residuals can for example arise from measurement errors, random shocks, or other unforeseen factors, such as a brief, storm-related grounding of flights.

In order to decompose a time series into these components, there exist techniques like the *Seasonal Decomposition of Time Series by Loess (STL)* [9]. This specific *time series decomposition (TSD)* approach uses a method called *Loess*, which stands for *Locally Weighted Scatterplot Smoothing* and iteratively fits smooth curves to subsets of the data, adjusting the weights of nearby points to capture local variations accurately. The output is an additive model $X_i = T_i + S_i + R_i$, representing each observation at time i as a summation of a trend, seasonal and residual component, denoted by T_i, S_i and R_i respectively.

2.1.2 Time Series Anomalies

In general, a single observation or a subset of observations can be defined as an anomaly if its behavior deviates significantly from what is expected to be normal [18]. Apart from the fact that this definition depends on the respective criteria for normal behavior, such a deviation can take various different forms.

For example, behavior may be deemed anomalous with regard to a singular variables expectation (temporal), or in the case of MTS, for usual correlations of multiple variables with one another (intermetric) or by violating both properties at the same time (temporal-intermetric) [30, 10]. For defining normal behavior one can also either look at the global or contextual behavior of the time series and differentiate between point anomalies and those which prevail over a longer period of time [6, 10]. While some definitions explicitly define anomalous behavior by dissimilarity of the time series components mentioned in Chapter 2.1.1 [10], other more general definitions just define it as a deviation with regard to some characteristic pattern [43].

Furthermore the detection relevance of different anomalies may depend on several factors, such as the application domain or the data quality, i.e., a local point extremum might contain relevant information or simply be a measurement error. Due to the varying approaches of different anomaly detection methodologies to model anomalous behavior and the associated strengths and weaknesses with respect to different anomaly types, knowledge about the domain as well as about the specific methods peculiarities is essential [43]. An example of this is shown in Figure 2.2, which compares the anomaly detection output of two models with varying approaches towards classifying anomalous behavior.

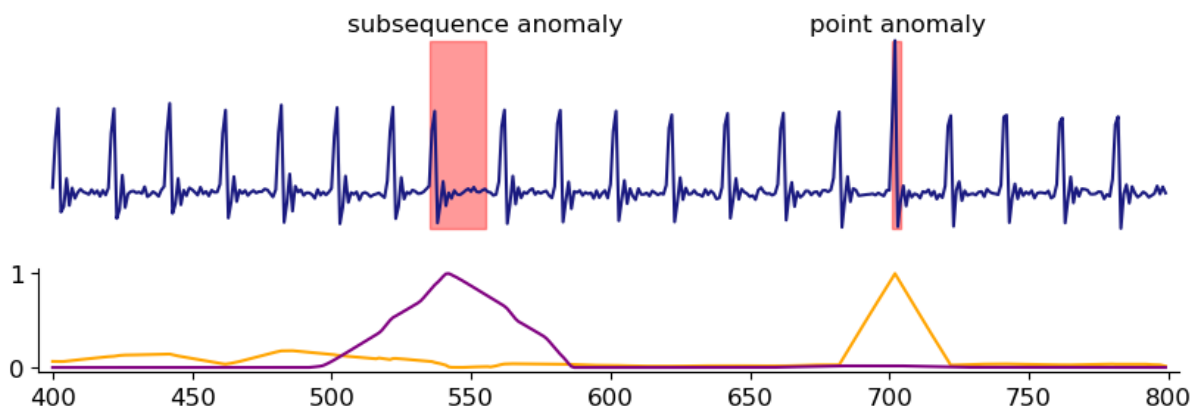


Figure 2.2: Plot depicting the result of applying two anomaly detection methods on a time series with a point as well as a subsequence anomaly. Due to the different quantification of anomalous behavior, neither method successfully detects both anomalies. Figure adopted from Schmidl et al. [43].

2.2 Anomaly Detection Methods for Time Series

Successful detection of varying anomaly types in complex time series continues to be an active area of research [43, 10, 6]. Although the binary differentiation of normal and anomalous instances can be understood as a classification task, it has some complicating peculiarities. For example, the by definition rare occurrence of anomalies, as well as their inhomogeneous character due to their already mentioned various potential manifestations.

Instead of just a binary label, methods may also assign an anomaly rank or even the exact extent of deviation in the form of an anomaly score, usually within a $[0, 1]$ interval, as in Figure 2.2. This further complicates the task, but, if successful, is accompanied by a significant increase in information regarding the absolute anomaly degree of data points as well as their relative differences [2].

Since anomaly detection resides mostly in the realm of semi-supervised or even unsupervised learning, *ground truth* in the form of anomaly labels during training is usually missing. This means that methods have to decide directly on the basis of observable structural properties of the data what characterizes a relevant deviation from normal behavior, without prior knowledge about what is desired by the user. This makes it particularly hard to determine with certainty to what degree a model learns correct assumptions during training [2].

2.2.1 Deep Learning

Deep Learning has emerged as a powerful paradigm for analyzing complex data, providing the ability to capture intricate temporal dependencies and automatically extract high-level

representations. Architectures based on fundamental models such as *Convolutional Neural Networks (CNNs)* [29] and *Recurrent Neural Networks (RNNs)*, for example *Long Short Memory (LSTM)* networks [20], have shown remarkable success on various time series tasks [21, 45, 7]. In particular, powerful new deep learning based approaches continue to be introduced with no end in sight.

A rather simple deep-learning architecture is the *Multi-layer-Perceptron (MLP)* [41], schematically shown in Figure 2.3. This is a fully-connected feed-forward network consisting of an input layer, an output layer and potentially any number of hidden layers, each consisting of neurons. Each neuron computes a weighted sum of the input and usually applies an activation function to allow for non-linear computations. So-called bias terms introduce a desired shift of this result as additional parameters independent of the current input. By optimizing the connection weights between neurons, a loss function that compares the deviation of input and output is minimized to learn an appropriate transformation.

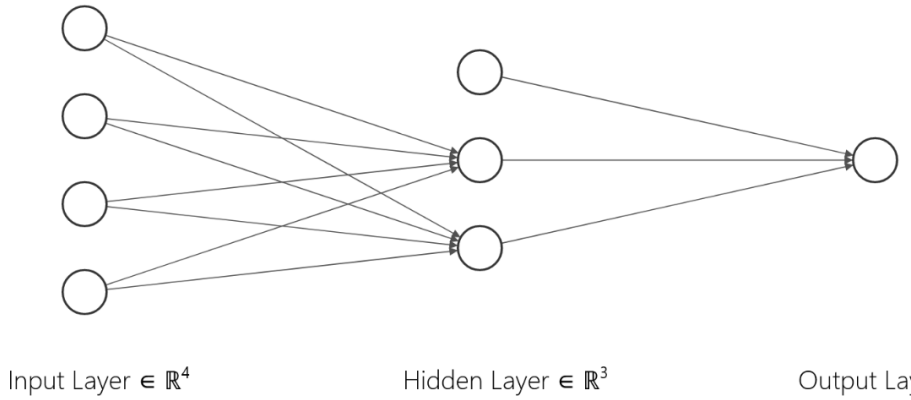


Figure 2.3: Schematic overview of a multilayer perceptron (MLP) [41] with 3 layers that maps a 4-dimensional input to a 1-dimensional output. The hidden layer contains a bias term which represents a learnable shift.

2.2.2 Ensemble Methods

A single high-performance model alone may not always provide optimal results because it may be sensitive to noise, exhibit overfitting, or be insufficiently robust in detecting rare anomalies [3, 2]. To address these challenges, ensemble methods have gained attention, combining multiple submodels to improve overall performance and increase the robustness of anomaly detection systems [44, 21, 25]. Ensemble methods can leverage the diversity of multiple models to capture different aspects of the data, reduce the biases of individual models, and improve overall accuracy and generalization ability [2].

In particular, due to the often limited prior knowledge in anomaly detection, ensemble methods can provide theoretically sound advantages in this regard with respect to the bias-variance tradeoff [2]. Common methods to increase the ensemble robustness as well as the diversity between submodels is to create different preconditions for them regarding considered instances or features. For example, *subsampling* can be used to consider different subsequences of the time series for individual submodels. In the case of MTS, *feature bagging* may also be employed to focus on only a subset of features in each submodel, thus promoting specialization and combating the effects of the so called *curse of dimensionality*, like increasing computational complexity and the concealment of relevant information [50].

2.2.3 Overview of Method Families

To tackle the problem of detecting anomalies in time series data, a broad variety of methods based on divers underlying anomaly quantification approaches have been proposed in the literature [43, 10, 6]. Although these methods can produce impressive results even for complex data sets and anomaly types, there still seems to be no universally best method, in particular when prior knowledge is limited [43]. In view of the heterogeneous nature of the conceivable anomaly types, it seems like a certain trade-off has to be made.

In addition to the differing complexity of a method and its construction as a singular anomaly detection model or a composition of these, a variety of different method families can be differentiated based on how their members quantify anomalous behavior [10, 43]. A few of these will be outlined in the following.

Distance-based [43]: Distance-based methods decide about the degree of anomalies by comparing points or subsequences of a time series (or mappings of them) based on specialized distance metrics. The underlying assumption is that a larger deviation from the usual distance is interpreted as anomalous. For the calculation of distances they may consider all subsequences of the time series, only those with a high degree of locality, or even proxies, e.g. in the form of cluster centroids.

Forecasting-based [10, 43]: The idea behind a forecasting-based method is to predict the future values of a time series using a so-called context window of historical data and to compare these predictions with the actual observed values. If there is a significant deviation between the predicted value and the actual observation, this indicates the presence of an anomaly.

Reconstruction-based [10, 43]: In reconstruction-based methods, the basic idea is to measure how successfully a transformed input can be reconstructed by the model. For this purpose, an input is first mapped into a latent space which, according to the model, is capable of preserving the structures characteristic of normal data. Following this assumption, a significant reconstruction error is indicative of an anomaly.

Encoding-based [43]: Similar to the reconstruction-based paradigm, encoding-based methods transfer time series sequences into a more concise representation, presumably focusing on the key features and characteristics of the time series, preserving relevant information while discarding unnecessary details. However, instead of reconstructing these latent representations, encoding-based methods assign anomaly scores to the sequences directly based on the properties displayed in this representation form.

Tree-based [43]: Tree-based methods are ensembles of several isolation trees (a so-called *Isolation Forest* [31]), which partition the data space using repeated splits of a random selection of features at random split locations. The number of splits required to isolate a given input then measures its degree of anomaly. This assumes that anomalies should be easier to isolate due to their unusual characteristics.

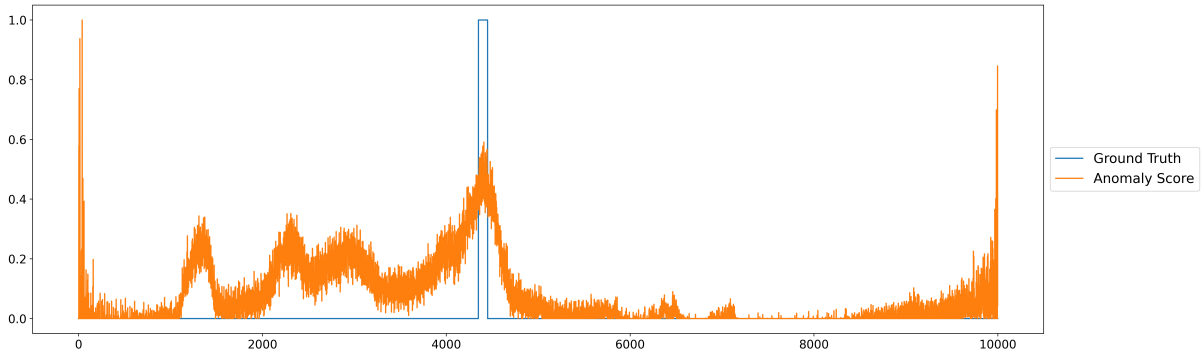
2.3 Performance Evaluation

The general baseline for measuring anomaly detection performance is the instance-by-instance match between the prediction and a ground truth of whether a sample is an anomaly or not. A prediction as normal (negative) or anomalous (positive) is hereby categorized as being either false or true, which can be represented in a so-called *confusion matrix* as shown in Figure 2.4. If the prediction of a method is not directly a binary decision, one or more thresholds can be considered for a corresponding mapping of ranks or scores to binary labels.

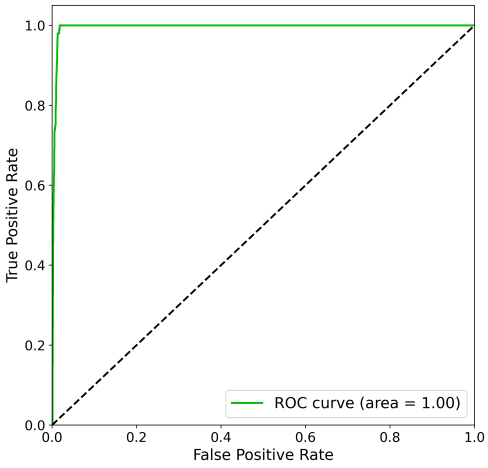
		Ground Truth		
		Anomalous (Positive)	Normal (Negative)	
Prediction	Anomalous (Positive)	True Positive (TP)	False Positive (FP)	Precision $\frac{TP}{TP+FP}$
	Normal (Negative)	False Negative (FN)	True Negative (TN)	False Omission Rate $\frac{FN}{FN+TN}$
		True Positive Rate (TPR) $\frac{TP}{TP+FN}$	False Positive Rate (FPR) $\frac{FP}{FP+TN}$	Accuracy $\frac{TP+TN}{TP+TN+FP+FN}$

Figure 2.4: Representation of an extended confusion matrix that visualizes the essential building blocks of anomaly detection performance quantification.

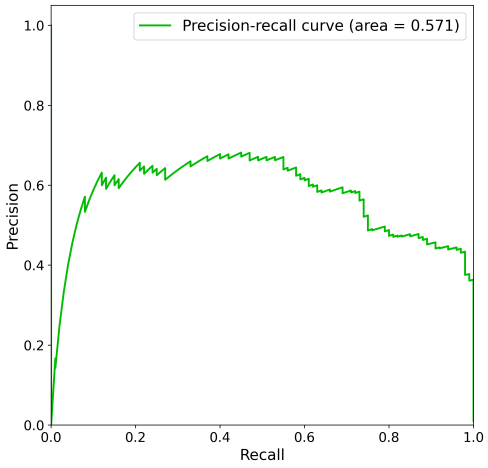
By comparing the absolute occurrences of the four resulting categories true positive, false positive, true negative and false negative, the quality of predictions can now be quantified in various ways. The two most common metrics for time series anomaly detection are probably the *area under the receiver operator curve (AUC-ROC)* [16] and the *area under the precision-recall curve (AUC-PR)* [37, 11].



(a) Anomaly detection method scores compared to ground truth.



(b) ROC curve of the result.



(c) PR curve of the result.

Figure 2.5: Side-by-side comparison of the ground truth and prediction of an anomaly detection method for a univariate time series, as well as two evaluation plots depicting the ROC and PR curves. Comparison of the near-perfect AUC-ROC score of 0.995 (rounded to 1.0 in the plot) with the AUC-PR value of 0.571 illustrates the higher accuracy requirements of the latter metric, which are not fully met due to the high anomaly scores for first and last time steps.

The ROC curve compares the true positive rate (TPR), also called recall, and the false positive rate (FPR) for different thresholds with respect to an underlying anomaly score. It thus visualizes the tradeoff between how many anomalies are actually recognized and how often a normal instance is falsely predicted as an anomaly. The AUC-ROC as the area under this curve evaluates the quality of this trade-off with a singular value.

The PR curve compares the TPR, or recall, with precision, i.e., the ratio of how often a prediction as an anomaly is actually correct. Given that the AUC-ROC does not explicitly consider precision, the PR curve is more informative regarding the accuracy of positive predictions, which may be more important when false positives are particularly

undesirable. Achieving a high AUC-PR is subsequently considered to be more challenging, given the characteristically class imbalance in anomaly detection due to the rare occurrence of anomalies and the corresponding difficulty of only precisely detecting those. This phenomenon is illustrated in Figure 2.5.

Furthermore, there are also metrics that explicitly consider the temporal nature of time series data. One such metric is the *area under the range-based precision and recall curve (AUC-P_TR_T)* [46]. This adapted version mitigates the strict requirements of the precision-recall curve by taking into account the order of scoring and thus adapting the measure to subsequences [43]. For example, partial overlaps of falsely predicted anomalies with real ones are not penalized as harshly. The assessment of importance between finding an anomaly and returning a largely matching overlap is also adjustable by dedicated parameters. While the choice of parameterization for this metric introduces an additional complexity component, it theoretically allows for fine-grained assessments based on use case-specific preferences and accounts for the sometimes difficult-to-identify start and end time points of subsequence anomalies.

3 DEAN: Deep Ensemble Anomaly Detection

This chapter provides a review of the initial version of the DEAN (Deep Ensemble Anomaly Detection) [25], the method on which the adapted version introduced in Chapter 4 and this thesis in general are based. After an introduction of the core methodology, the novel interpretation approach developed for this method is presented and the empirical findings obtained in the original paper are summarized and contextualized for later comparison in Chapters 5 and 7.

3.1 DEAN Methodology

As the name suggests, DEAN combines neural networks, or more precisely comparatively weak MLPs, in an ensemble approach to detect anomalies. For this purpose, several fairly constrained, architectural homogeneous models are first trained in parallel from each other. Diversification between the models is achieved by a simple loss function in combination with the enforced constraints and differing preconditions. Subsequently, the individual submodels are combined with equal weighting to get the final outlier score.

To ensure a well-functioning deep ensemble-based anomaly detection method some essential properties should be satisfied. For this purpose, the authors of DEAN suggest the following four *DEAN properties* which are deemed to be beneficial regarding the generation and characteristics of submodels as well as for their combination and are consequently upheld by the DEAN method.

Scalability: Generating the individual submodels (and their combination) should have a reasonably benign runtime behavior. The fast generation of numerous models in particular enables the achievement of ensemble-typical properties such as the reduction of variance and the associated high robustness.

Depth: The individual submodels are not required to perform highly competitive on their own, but should be able to learn complex enough interrelationships so that their combination can identify non-trivial anomalies.

Variability: While the variance of the final ensemble should be low, the opposite is true for the variance between models. In order for the combination of a larger number of submodels to add value, they must learn different characteristics; accordingly, a sufficiently high inter-model variance is necessary.

Consistency: For the final combination, the individual models should provide consistent, comparable results and fit well to the combination function. In particular, the same scaling should be used for the output and an arbitrary weighting of different submodel contributions should be avoided, especially with regard to the subsequent interpretabiltiy of the overall ensemble prediction.

3.1.1 Submodel Design

The learning objective of each submodels neural network f is to uncover one of assumed many nearly constant relations $g(x) = k \cdot (1 \pm \delta)$ in the training data input vectors $x \in X_{train}$. If the learned relation is valid for the overall data distribution, this means $f(x') = \frac{g(x')}{k} \approx 1$ should hold for most unobserved input vectors $x' \in X_{test}$. By using an ensemble, the individual submodels can naturally specialize to one of these relations each. Significant violations of these relations with respect to the entire ensemble are then considered indicative of an anomaly.

To accomplish this learning objective, each neural network minimizes the novel loss function given in Equation 1, namely the squared distance of its output to a constant 1. This corresponds to minimizing δ^2 in our relation g, so that f learns the more constant and therefore presumably most meaningful relations across the training inputs $x \in X_{train}$.

$$loss_{DEAN} = (f(x) - 1)^2 = \left(\frac{k}{k} \cdot (1 + \delta) - 1\right)^2 = \delta^2 \tag{1}$$

While the loss functions simplicity allows for an easy training process and due to the various potential solutions seems to encourage both scalability and variety, it also enforces certain restrictions to avoid degenerated models like $f(x) = 1$. With regard to the activation function, this means for example the unsuitability of options like the sigmoid function $a(x) = \frac{1}{1+e^{-x}}$, because this way the model could trivially learn a mapping to 1 irrespective of the provided input. A visualisation of this problem is given in Figure 3.1. To combat this, the rectified linear unit (ReLU) activation function $a(x) = max(x, 0)$ is being used for the hidden layers.

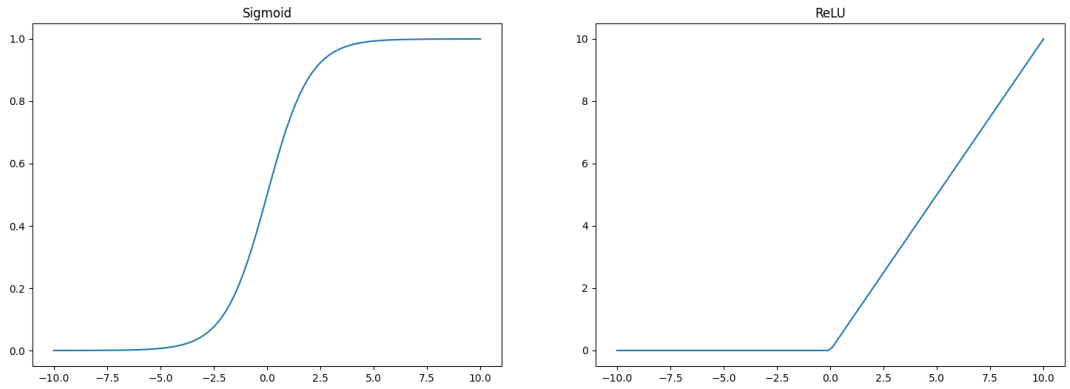


Figure 3.1: Comparison of the functions ReLU and Sigmoid for the input range $[-10, 10]$. The rapid convergence of the sigmoid function towards 1 for positive values illustrates the problem when used in combination with the loss function of DEAN.

Including the usual bias term as a learned constant shift also increases the risk of trivial solutions. The authors note that while partially removing the bias theoretically makes the model no longer be able to universally approximate functions, it also often leads to faster convergence and did not significantly change the ensembles performance in the conducted experiments, therefore not necessarily violating the depth property. According to the variability property, one could argue that a singular model should not be too powerful anyway, as this tends to limit the variety and thus the benefits of using ensemble techniques in the first place.

After the training is completed, a DEAN submodel predicts the anomaly degree of an unseen input $x \in X_{test}$ by the Equation 2, namely the absolute deviation between the neural networks output for the given input and its mean output when applied to the training data.

$$p(x) = |f(x) - q| \; with \; q = \frac{1}{|X_{train}|} \sum_{x \in X_{train}} f(x) \tag{2}$$

In terms of preprocessing, the input data values are normalized to an $[0, 1]$ interval. Furthermore the authors explored mapping the input to latent spaces via a combination of multiple autoencoders.

3.1.2 Ensemble Composition

A random weight initialation for the neural networks paired with the simple loss function may already lead to a certain degree of variation between the submodels. To further this variety, early stopping and feature bagging [28] are employed, with the former ending the training process if no improvement was observed for a certain time frame and the later leading to each model only considering a limited number (bag) of features each. Another advantage of feature bagging is the positive impact on the runtime due to the lower input dimensionality as well as generally limiting the impact of the curse of dimensionality [50].

The combination of multiple DEAN submodels is done by the formula given in Equation 3. This calculation of the final ensemble score is quite similar to the root mean square (RMS), with the only difference being that the division by ensemble size n is done after taking the square root of the summed up squared individual submodel predictions p_i, not before. With each submodel using the same normalized input and taking the same number of features into account and equal weighting, this confirms with the consistency property.

$$score_{DEAN} = \frac{1}{n} \sqrt{\sum_{i=1}^{n} p_i(x)^2} \tag{3}$$

3.2 Interpretability by Feature Importance

The potentially large number of models, as well as the networks internal complexities, seem to imply poor a interpretability of DEAN. However, an analysis of the ensemble structure, more specifically correlations between the predictions of individual submodels and their considered features, yields an interesting form of interpretability. This approach quantifies the importance of specific features in general based on their impact on individual scores.

To be precise, a feature f^* is considered important with regard to a given data point, if using it significantly alters the ensemble score. This is measured by observing the difference of the average submodel predictions p_i where f^* is part of the submodels feature set F_i^* to the overall average prediction. A large absolute value corresponds to an important feature due to the associated large influence, with a positive value indicating that consideration of the feature tends to favor higher anomaly prediction and vice versa. For a slightly adjusted formula to calculate this score compared to the original publication, see Equation 4.

$$I^{f^*} = \frac{1}{|S^{f^*}|} \sum_{i=1}^{|S^{f^*}|} S_i^{f^*}(x) - \frac{1}{n} \sum_{i}^{n} p_i(x) \; with \; S^{f^*} = \{p_i | f^* \in F_i^*\} \tag{4}$$

The authors of DEAN note, that in order to generate an accurate *importance score* I^{f^*}, it is imperative that the number of models using a feature S^{f^*} has to be much larger than the general bag size, with $bag = |F_i| \;\; \forall \, 1 \leq i \leq n$. This is necessary, because otherwise the randomness of feature bagging tends to dominates the importance score. In particular it holds that the overall ensemble size should far exceed the total number of feature available for this interpretability approach to work.

3.3 Empirical Findings

As part of DEAN's original publication, benchmark experiments were performed on the CIFAR-100 [27] and MNIST [12] datasets, as well as further experiments, for example regarding ensemble structure and interpretability.

In terms of anomaly detection performance, the DEAN versions with and without bias performed best in terms of average observed ROC-AUC value compared to alternative models such as *Autoencoder* [42] and *Isolation Forest* [31]. Since DEAN's runtime efficiency scales directly with the number of models, a comparison was made based on the ensemble size needed to achieve this level of performance. Runtime-wise, DEAN lagged behind the very efficient Isolation Forest, but generally performed also quite well.

The experimentally observed fast ensemble convergence to a ROC-AUC score of > 0.95 with about 30 submodels on parts of the MNIST dataset suggests an adequate individual

performance of the models for comparatively complex tasks and shows that a manageable ensemble size can already outperform more sophisticated methods.

Regarding the parameterization, no relevant performance differences were found for the DEAN variants with and without bias. Furthermore, it is indicated that a variation of the other hyperparameters like bagging size, model depth, activation function and so on (in reasonable scales) did not have a large influence on the performance either.

For the experiments with regard to the interpretability, the bag size was downscaled and the bias term was omitted, so that the number of necessary models remained manageable. Especially on the MNIST data set the experiments delivered intuitive interpretations for the importance of single features (pixels) for the overall anomaly quantification.

4 Adaptation of DEAN for Time Series

Following the introduction of DEAN in the last chapter, we now proceed to the design of DEAN-TS as an adapted version for time series anomaly detection. We begin with preliminary considerations in the form of identified objectives and associated challenges. This is followed by an outline of the core DEAN-TS methodology and subsequently the potential of integrating time series decomposition as an additional preprocessing step to improve performance and interpretability.

4.1 Objectives and Challenges

Our main objective is to explore the suitability of the DEAN core concepts for the time series domain, that is, employing an ensemble of constrained MLPs to discern diverse near-constant relations in the data by minimizing the DEAN loss function. For this we want do devise an adapted methodology, taking into account the peculiarities of temporal data and evaluate its competitiveness with regard to the following three criteria:

1. Anomaly detection quality
2. Runtime efficiency
3. Interpretability of predictions

The anomaly detection quality should be robust and cover various types of anomalies with the runtime efficiency being seen in the context of an suitable ensemble size to accomplish this task. With regard to the interpretability of predictions, we want to adapt the novel approach of the original DEAN method. For accomplishing these objectives it seems beneficial to largely retain the DEAN properties introduced in Chapter 3.1. Additionally, there are several potential challenges to consider.

A way must be found to model the assumed potentially complex dependencies of successive observations, where both contextual temporal and inter-metric information may be required, despite the simple MLP architecture. Based on the ensemble approach, at the same time the individual submodels must be diverse, focusing on different underlying relations. Accordingly, a compromise must be found between the necessary complexity of the individual models and the resulting unfavorable development with respect to inter-model diversity as well as runtime demands. The submodel variability should also not be at the expense of the applicability of DEAN's interpretability method.

4.2 DEAN-TS Methodology

In order to meaningfully test the suitability of DEAN for time series, the basic submodel architecture is retained. A single submodel still consists of a comparatively shallow MLP which minimizes the DEAN loss function to learn nearly constant relations, thus trying to map each input to a constant 1 value. Because of the loss functions simplicity, partial-constant activation functions are also still omitted. Accordingly, the evaluation of a new data point by a submodel continues to consist of the distance between the network's associated output and the mean output q it produced during training. Since this approach (still) directly attributes the anomaly score to an encoded representation of the input, namely the network-derived value, which is hopefully close to 1, DEAN-TS can be classified as encoding-based according to the categorization from Chapter 2.2.3.

Given the mentioned consistency of the submodel architecture, the adjustments made to derive the adopted DEAN-TS methodology in term focus on the suitable selection and preprocessing of the input, meaningful diversification between the submodels and their subsequent combination, all under consideration of upholding the DEAN properties.

As a general preprocessing step, we first standardize the time series training data and use the mean and standard deviation value observed here for the same scaling when performing anomaly detection on unseen time series. Each submodel processes as input the observations of several time steps of the time series simultaneously by applying a window-based approach. This seems necessary to preserve the **depth** property, because MLPs, in contrast to recurrent networks, cannot otherwise directly represent the temporal dependencies between successive observations. At the same time, by avoiding more complex techniques of integrating temporal reasoning, we do not violate the methods **scalability** to a relevant degree.

Besides varying window properties and in the case of multivariate time series by applying feature bagging, the submodels can be further diversified to promote **variability** by employing subsampling. Considering only a subset of the time steps during training has a positive impact on runtime and may even help the performance in an unsupervised setting, given that an anomaly is no longer considered in all of the submodels to learn normal behavior and the associated relations.

For combining the submodel scores, we propose an alternative but still **consistency** ensuring combination function, which tends to separate anomalous from normal data more clearly and, moreover, represents a compromise between advantageous bias and variance properties of alternative combination functions [2].

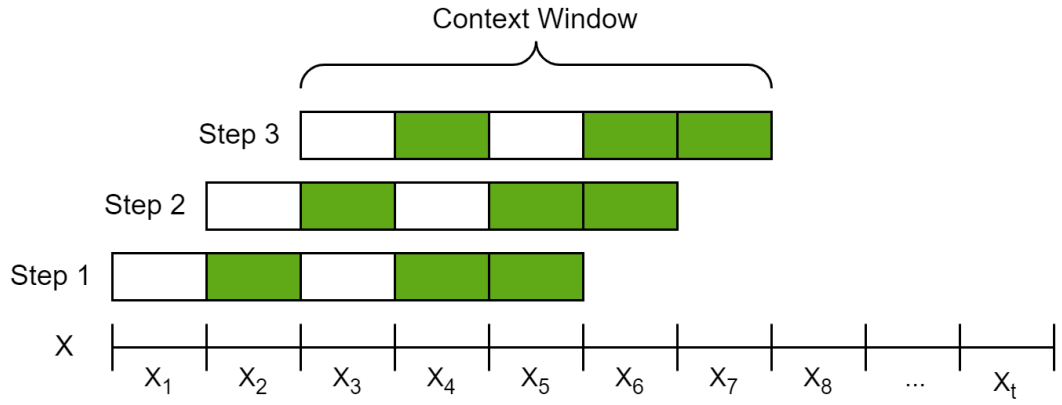


Figure 4.1: Representation of the input generation through a context window with $look\_back = 4$ and $lag\_indices = [1, 3]$. Accordingly, the input vector (X_2, X_4, X_5) would be processed in the first training or prediction step.

4.2.1 Adjustments for Temporal Data

Given the assumed temporal continuity of time series data, each DEAN-TS submodel processes during training and for predictions not only singular points as input, but subsequences of time-ordered observations with a given dropout. This is done by sliding a context window with a stride of one across a given time series $X = (X_1, X_2, ..., X_t)$. For each submodel, the context window length and dropout configuration are set individually, but in such a way that each submodel processes inputs of the same length, i.e. has the same *bag* value.

An alternative interpretation of this approach is that, given the context window width w, we prepend a subset of the last $w - 1$ observations to each observation X_i with $i \geq w$ to build the corresponding input vector. In this interpretation, $w - 1$ is specified by the *look_back* configuration parameter, and the subset of previous observations by a list of indices, the *lag_indices*. These *lag_indices* specify the shift in time steps relative to the current observation X_i to identify the additional elements of our input vector. In our initial interpretation, this corresponds to those elements of our context window that are not dropped.

Then, for each time step i, a time-ordered input vector is generated and processed with $look\_back < i \leq t$. A simple example of this procedure is shown in Figure 4.1. For multivariate time series, the same *lag_indices* are used to determine the observations for each of the variables of interest, allowing the submodel to learn inter-metric relationships as well. If multiple variables are considered, the vectors are concatenated and the result is flattened accordingly, resulting in a general vector of length $|lag\_indices + 1| \cdot d = bag$ for d considered variables.

Due to the overlapping context windows, observations from most time steps are included in multiple input vectors. To obtain individual anomaly scores for each of these observations, the outputs of all context windows that include it are averaged. An inherent disadvantage of reverse-window-based approaches like this is the imbalance due to the fact that the last and first $|look\_back|$ many time points are considered less frequently than the remaining ones. While for a relatively small window size this should only affect comparatively few observations to a significant degree, this has to be taken into account when evaluating the reliability of the corresponding scores.

4.2.2 Submodel Diversification

To diversify the submodels, they apply context windows with different properties for preprocessing the input. In order to learn relations relevant for anomalies occurring over different time lengths, we first randomly and uniformly select a $look\_back$ value from a pre-specified integer interval of values, the so-called $look\_back\_range$. Then we randomly select the list of $lag\_indices$ from the integer interval $[1, look\_back]$ without replacement. The number of selected $lag\_indices$ remains constant, so larger context windows are necessarily accompanied by a higher dropout rate $\frac{1-|lag\_indices|}{look\_back}$. This promotes scalability while keeping the variability between submodels high.

With a sufficiently high $look\_back\_range$ value, this allows longer time periods to be considered. At the same time, this approach leads to a bias in favor of smaller lag indices. On the one hand, this is due to the already mentioned higher dropout rate for larger $look\_back$ values, but also to the fact that drawing a low $look\_back$ value only allows to consider nearby time steps. Therefore, it can be expected that more models will focus on relationships that have a close temporal proximity to the initial observation corresponding to the last value in the input vector. Given the assumption that time series usually show strong temporal continuity, this bias is deliberately introduced.

In addition to early stopping, we also use feature bagging for multivariate time series. Considering only a selection of variables in each submodel helps to focus on different inter-metric relationships and improves runtime efficiency and distinguishability of data points by mitigating the so-called curse of dimensionality [50]. Given the fact that we already consider multiple features for the univariate case because of the window-based input, while theoretically the bag size still scales only linearly with dimensionality, in practice this can have a significant positive impact.

Complementary to the selection of a subset of variables, the time steps considered per model can also be limited by means of subsampling. Due to the assumed strong temporal continuity, continuous sub-ranges of the time series are selected for the submodels. Besides additional runtime benefits, this may also lead to performance gains, in particular for

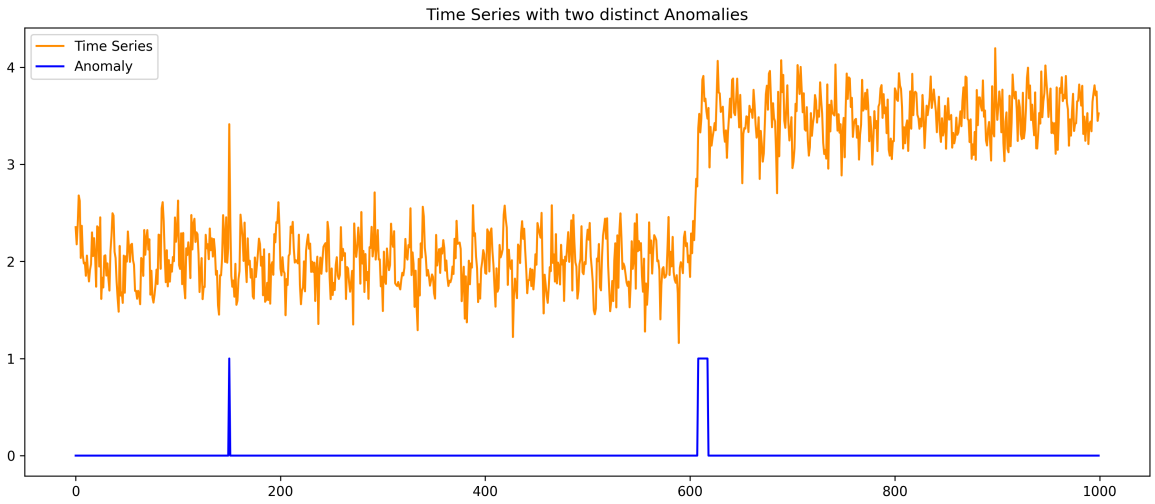


Figure 4.2: Example of a time series with two distinct anomalies and a comparatively large time interval between them. Subsampling may help anomaly detection performance by not always incorporating both anomalies when learning normal behavior patterns.

unsupervised learning, where we cannot guarantee the absence of anomalies when learning normal relationships. As visualized in Figure 4.2, considering a subsample per submodel avoids incorporating these anomalies into each learned relation.

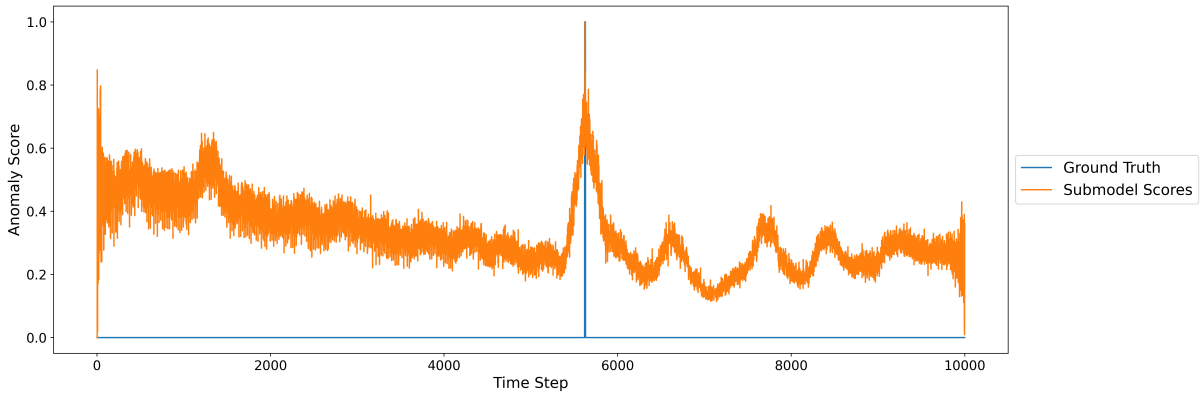
In general, it is conceivable to select these subsamples randomized or structured with respect to the starting point as well as the subsequence lengths. Notably, smaller sample sizes increase the runtime efficiency, but also the risk of ignoring relevant information and not learning the latent relationships robustly enough. Given that different sample sizes may also generally work better depending on the specific preconditions, it seems reasonable to vary the considered sampling rate. This can also be seen as an implicit exploration of an ideal environment for the submodel in absence of ground truth [2].

For a randomized approach, this may be done by specifying a random subsampling range rs_range, which defines a lower and upper bound for the sample size, and randomly selecting the first considered time steps afterwards. This approach will subsequently be called *random subsampling*. While this method eliminates human bias, it also risks completely ignoring information and introduces random implicit weighting of time series regions.

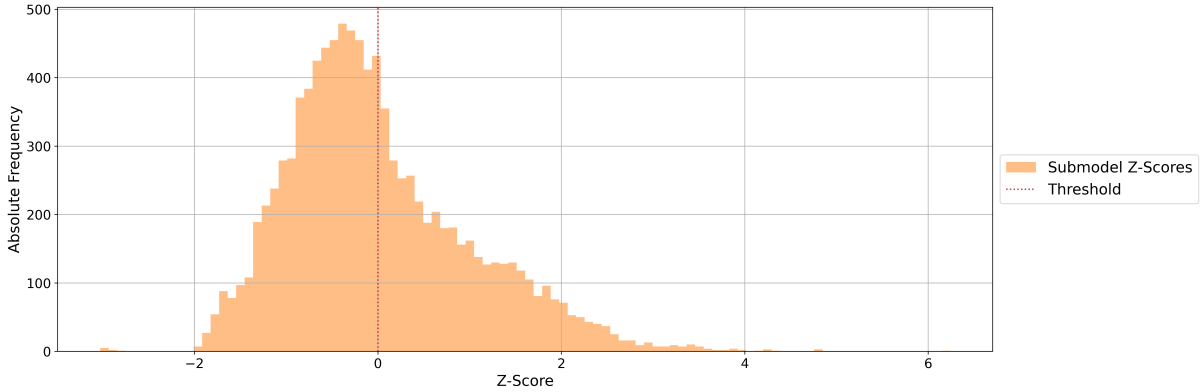
We therefore propose using a *structured subsampling* approach, that is, using different sample sizes, but overall considering all time steps the same number of times. For this purpose, several relative values $(r_1, ..., r_k)$ are selected in advance, each of which determines a sample size of $\lceil r_i \cdot t \rceil$ to split a time series of length t into roughly $(1/r_i)$ consecutive,

contiguous subsamples that together cover the entire time series. Subsequently, for each r_i we specify a quantity m_i, so that if $t/\frac{1}{r_i}$ is an integer value, $m_i \cdot (1/r_i)$ many submodels are trained on non-overlapping subsamples of the given split size, in total considering the complete time series m_i times. For example, given $t = 100$, $r_i = 0.25$, $m_i = 5$, this means to a total of 20 trained submodels with 5 of them considering each quarter of the time series.

4.2.3 Outlier Score Combination



(a) Submodel predictions next to the ground truth identifying the anomalies.



(b) Distribution of the z-scores corresponding to the submodel predictions.

Figure 4.3: Example of converting the predictions of a single submodel into z-scores. The result roughly follows a normal distribution, with the high values for the first and last predictions leading to some positive valued outliers. A threshold of 0 already leads to more than half of the submodel predictions not being considered.

The default method of DEAN-TS for combining the individual submodel predictions is based on the so-called *tresh method* [2]. In this combination approach, the predictions p of each submodel are first converted into z-scores by calculating $z = \frac{p-\mu_p}{\sigma_p}$ using the observed mean μ_p and standard deviation σ_p over all of its predictions. The resulting

z-scores correspond to the number of standard deviations that the associated inputs prediction deviates from the submodels mean prediction. An illustration of the effect this transformation has can be seen in Figure 4.3.

A threshold is then set and the values above it are summed over all submodels, resulting in a single prediction value for each input. The specified threshold value thereby controls the degree of punishment for appearing as a normal instance in some of the learned relations; for example, the negative influence of all below-average submodel predictions would be completely removed if the threshold were set to 0, which is the value Aggarwal et al. suggest [2]. Ensuring that the final anomaly score of the ensemble lies within the usual interval of $[0, 1]$, DEAN-TS afterwards normalizes these values. The overall ensemble scoring procedure is summarized by Equation 5, with ϕ applying the min-max normalization.

$$score_{DEAN-TS} = \phi\left(\sum_{i=1}^{n} z_i(x) \cdot \Theta_{thresh}(z_i(x))\right) \; with \; \Theta_{thresh}(a) = \begin{cases} 1 & \text{if } a > thresh \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

One could argue, that the thresh method is a more balanced type of score combination compared to the two commonly used methods of either choosing the averaged or maximum prediction [2]. Mitigating the importance of sometimes getting a low anomaly score shares similarities with selecting the maximum, but is more resistant to fluctuations. This increased resilience is achieved by considering multiple values, similar to classical averaging or the dean combination method. At the same time, a preference regarding the trade-off between a high-risk emphasis of singular high predictions and robustness can be controlled by the chosen threshold.

The balanced nature of the thresh method can be further illustrated by the following example. If a data point x were to be classified as absolutely normal ($p(x) = 0$) based on half of the learned relations and maximally abnormal ($p(x) = 1$) by the other half, it would achieve a significantly higher ensemble score using the thresh method than a data point that is always assigned a score of 0.5. When using averaging, however, both data points would be assigned the exact same ensemble score. At the same time, unlike when selecting the maximum submodel prediction, the tresh method theoretically still permits achieving an even higher ensemble score, for example when all submodels would assign a score of 1.

Notably, when using the tresh method experimentally with a threshold of 0, a clearer distinction between abnormal and normal data points as well as a deflation of the average score could be observed compared to the DEAN combination method and the aforementioned alternatives. This effect is shown in Figure 4.4. While the difference in AUC-ROC

performance was negligible in this particular case, the clearer distinction may help a human observer in his analysis.

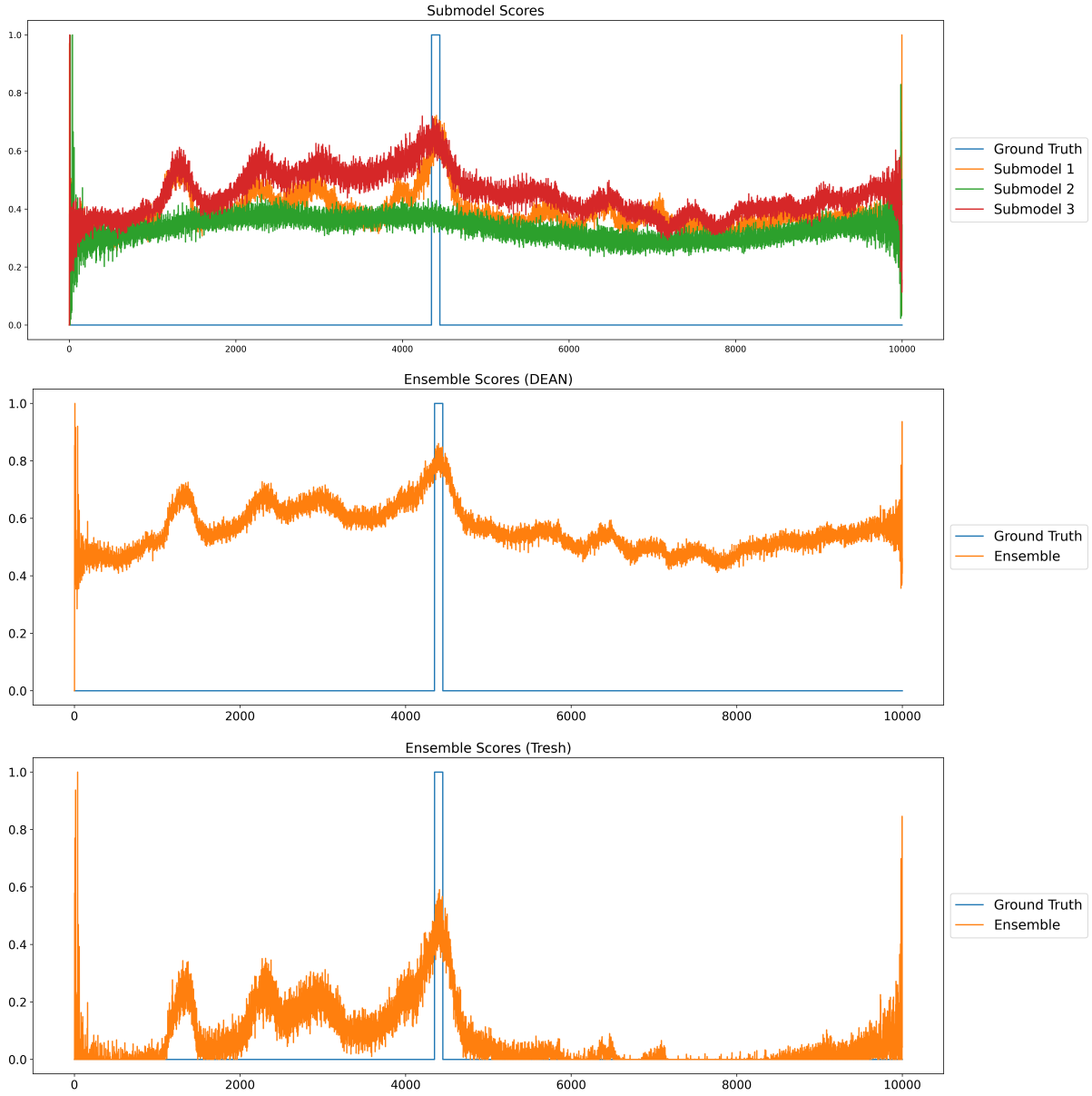


Figure 4.4: An example comparison of ensemble scores based on the same three submodels using the thresh method (treshold=0) compared to the original DEAN approach for score combination. The average ensemble score is significantly lower when using the thresh method and the value range is better exploited. A similar effect can be observed when comparing the thresh method with maximization or averaging.

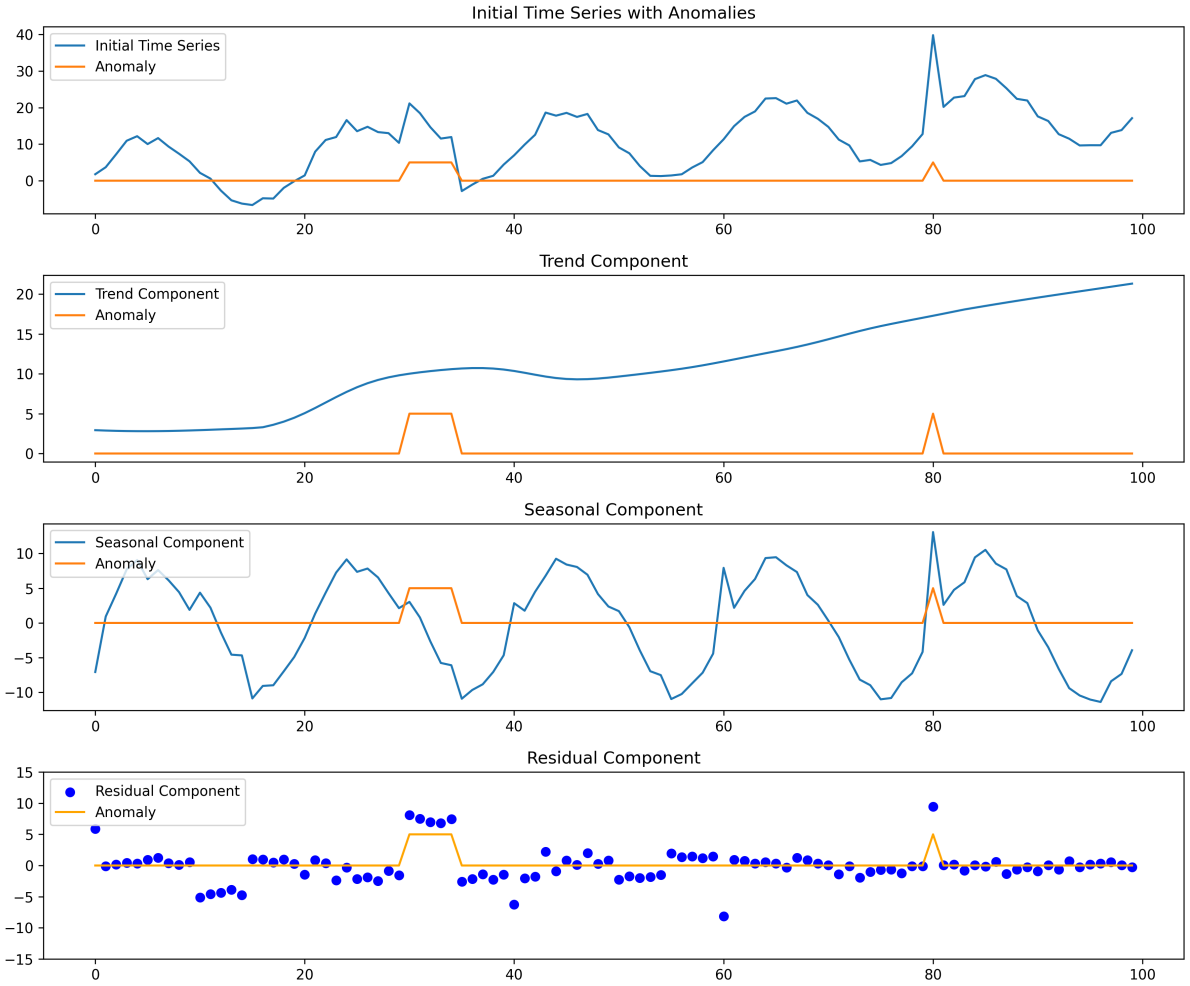


Figure 4.5: Example of a time series decomposition using the STL method. The first anomaly appears to be anomalous when considering the combination of trend and residuals, while the second anomaly has a locally distinct residual value and is also quite prominent in the seasonal component.

4.3 Preprocessing via Time Series Decomposition

While it is possible, as described so far, to directly process single or multiple of the observed data points during anomaly detection on time series, more complex preprocessing steps can be applied in addition to simple standardization and normalization to transform the input into a potentially more suitable representation. Related methods for this purpose use, for example, different variants of autoencoders [45, 44], discretization and subsequent partitioning of the values using clustering [34], or an explicit decomposition of the time series into diverse structural components [14, 47], similar to those introduced in chapter 2.1.1. While the experimental evaluation of DEAN has relied on a combination of multiple autoencoders in some experiments [25], for DEAN-TS we propose the integration of time series decomposition (TSD) as a natural preprocessing step for time series.

One possible approach is to use the STL method introduced in chapter 2.1.1 to additively decompose each variable X_i of a time series X into a trend component T_i, a seasonality component S_i, and a residual R_i, respectively, in the form $X_i = T_i + S_i + R_i$. An example of applying the STL method to a univariate time series with two different anomalies is shown in Figure 4.5. Both anomalies are now relatively easy to detect in two of the components, especially when considered simultaneously. This way of splitting the time series into semantically expressive aspects in combination with feature bagging potentially makes it easier for DEAN-TS to learn diverse, relevant and robust relations by individual submodels in order to be able to detect even complex anomaly types.

When using the STL method, the structural properties of the time series should be consistent with the assumptions of the method. In particular, a certain smoothness of the trend as well as seasonality is assumed, so an application to time series with a more chaotic basic structure is likely to be less successful. Ideally, the time series should be stationary, but at least some periodicity is expected. The existence as well as the frequency of this periodicity can be estimated in advance, for example, by analyzing the *Autocorrelation Function (ACF)*.

4.4 Interpretability of DEAN-TS

Predicting a data point to be anomalous without explaining why can significantly limit the usefulness of a method [21]. Accordingly, several approaches have been developed to facilitate the interpretability of anomaly predictions, for example, based on the contribution of individual features [25, 45, 35] or time steps [35]. In principle, the methodology of feature importance based interpretability of DEAN introduced in Chapter 4.2 can be directly applied to DEAN-TS (see Chapter 3.2). The features importance score I^f for individual predictions would be assigned to each lag index f in a completely analogous way. Besides the necessity to train a disproportionate amount of submodels compared to the number of features, the variable window width of DEAN-TS further complicates the practical application.

With the integration of time series decomposition, as proposed in Chapter 4.3, an alternative interpretation of this interpretability approach becomes possible. With the TSD components as a very limited feature set, it becomes incomparably easier to train an adequate number of submodels for the DEAN interpretability methodology. Furthermore, since the choice of components to be considered is randomly selected with equal probability by feature bagging with a range of one to three, there is no implicit weighting as in an application to lag indices. In addition, the importance of the time series components for individual predictions provides an immediately understandable rationale.

Since the concrete interaction of the individual TSD components provides additional

information, we define an importance score I^{TSD} for DEAN-TS based on the evaluation of a combination f^{TSD} of these components at a time. Formally, this yields the Equation 6, which calculates the difference between the prediction of those submodels that consider this combination as a feature set and the average prediction of all submodels. As with DEAN, a larger positive value indicates a high degree of anomaly with respect to that component combination, which is therefore considered important for that data point. For those data points predicted by the ensemble to be particularly anomalous, an explanation for the prediction can be provided in this way. To generalize the importance score of a data point to a subsequence of the time series, one can simply average the importance score of the TSD component combination for all affected data points.

$$I^{f^{TSD}} = \frac{1}{|S^{TSD}|} \sum_{i=1}^{|S^{TSD}|} S_i^{TSD}(x) - \frac{1}{n} \sum_{i}^{n} p_i(x) \; with \; S^{TSD} = \{p_i | f^{TSD} = f_i^{TSD}\} \quad (6)$$

To illustrate this rationale, we can again consider Figure 4.5. The anomaly that occurs first would be expected to receive a higher anomaly score than the average ensemble score from submodels that simultaneously include the residual and trend components. Since the entire anomalous region is clearly identifiable, especially in the case of the residual component, it can be realistically assumed that all or part of the anomaly will be recognized as such. The later anomaly, on the other hand, is only clearly noticeable in the affected time step for both the seasonality and residual components, so that a high importance score is expected here for the corresponding combination. By analyzing the relation of specific importance sore combinations to such individual anomaly types, meaningful high-level explanations may be possible.

5 Benchmark Evaluation (Semi-Supervised)

In this chapter, we experimentally investigate the performance and runtime efficiency of DEAN-TS for anomaly detection in a semi-supervised learning environment. Semi-supervised here means that training is performed on a time series that is structurally similar to the test time series, where the training data is unlabeled, but certainly does not contain anomalies, in order to facilitate learning of normal structures. For this purpose, several slightly different versions of the methodology presented in chapter 4.2 are compared with competing methods from related work on a variety of real and synthetic time series with differing characteristics regarding both normal behavior and anomaly types.

5.1 Environment and Setup

The following experiments were performed using the *TimeEval* [48] time series anomaly detection benchmark tool and are structurally similar to those in a detailed evaluation paper by its developers [43]. In particular, we use the same synthetic time series datasets generated by the time series generation tool *GutenTAG* [48] and a selection of the real time series datasets that were being considered. TimeEval promotes a fair comparison of methods by restricting the allowed resources, measures the pure execution time of the algorithms broken down into training and testing time, and provides all the performance metrics introduced in Chapter 2.3, namely AUC-ROC, AUC-PR and AUC-P_TR_T, with the latter parameterized according to the defaults proposed in its original publication [46], namely $\alpha = 0$, $\gamma = 1$ and a flat bias. While all these metrics were considered for the average performance across all datasets in a given dataset collection, the analyses of individual subgroups from the GutenTAG datasets focus on the averaged AUC-ROC score to keep the amount of information manageable, similar to the original survey [43] and DEAN publications [25].

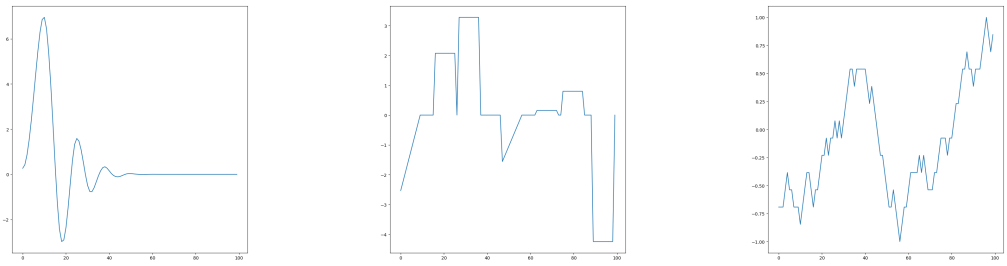
Execution environment: As an execution platform, a computer cluster running Ubuntu 20.04.5 LTS with x86-64 architecture, an Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz with 40 cores and 256 GB of RAM was used. The performance of this cluster was intentionally underutilized by limiting it to 8 CPU cores and 32 GB of memory to ensure reasonable equal constraints. We also limited the maximum runtime to 20 minutes for the GutenTAG and NASA datasets and 30 minutes for the more demanding SMD dataset collection.

Error handling: Some of the experiments terminated with an error for various reasons. The exact number of errors per algorithm is given once for each data set collection at

the beginning of the respective evaluation in a summary table, as well as a corresponding explanation of their causes, i.e. whether these errors occurred on of the algorithm, were a violation of the resource limit, or something else. For a fair assessment of performance, failed runs are initially assigned a value of 0 when averaging individual metrics in the tables. To make this relatively strong penalty more transparent, a "+" is added after each performance value to indicate the amount by which the averages increase when the failed runs are ignored. To avoid overly distorting the box plots, the erroneous runs were ignored in their visualization.

5.1.1 Benchmark Datasets

The datasets generated by GutenTAG [48] allow an in-depth analysis of the performance capabilities of DEAN-TS. In the following experiments, 193 diverse synthetic datasets were used, each consisting of a training and a test time series. Each of these training and test time series has a length of 10000, a dimensionality varying between 1 and 20, and exhibits one of 5 possible base oscillations combined with one or more anomalies. Besides a basic *Sine Wave (Sine)* and simple *Polynomial (Poly)* oscillations, there are also more unusual shapes with exemplary plots shown in Figure 5.1. The *Electrocardiogram (ECG)* oscillation mimics a repeated characteristic ECG curve, each time followed by a plateau. Finally, the more chaotic *Cylinder Bell Funnel (CBF)* [23] and *Random Walk (RW)* oscillations exhibit the most structural complexity. The exact parameterization used to generate a time series of the same base oscillations, such as the underlying noise and frequencies, is not uniform to increase the range of structural variation.

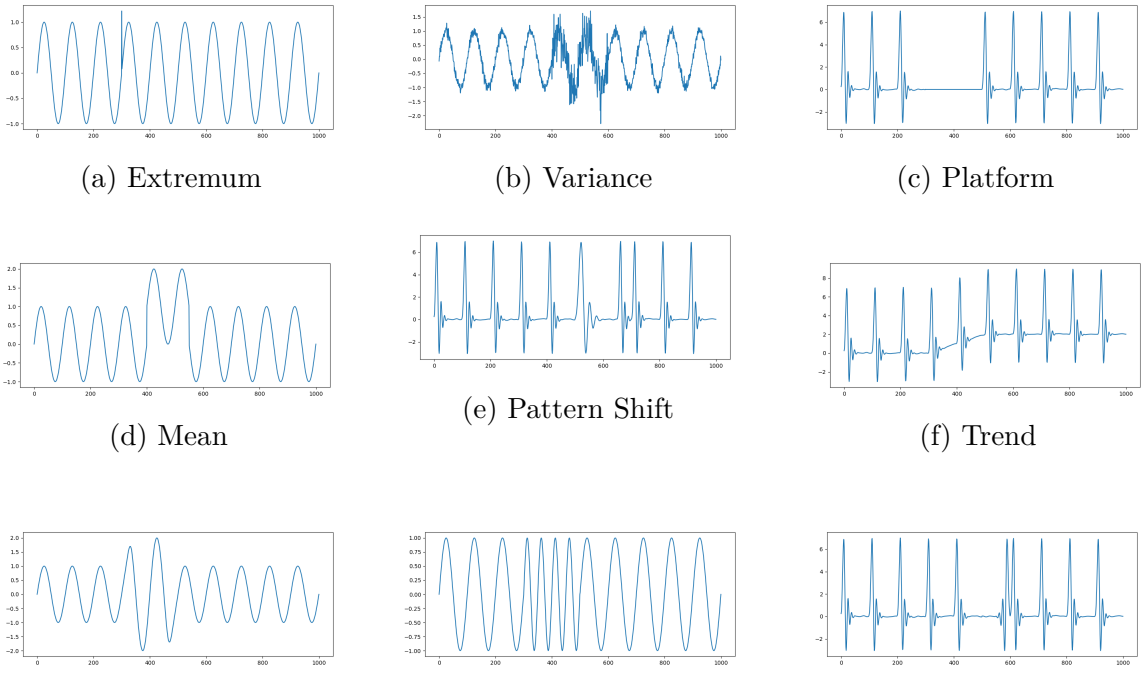


(a) Electrocardiogram (b) Cylinder Bell Funnel (c) Random Walk

Figure 5.1: Examples of the three more complex base oscillations.

Given the semi-supervised setting of the experiments performed, the same base oscillations, with partly different properties, are used in the respective related training and test time series. The corresponding test sets are injected with one or multiple anomalies of 9 different types, each of which is exemplified by the overview in Figure 5.2. Some introduce general anomalous artifacts into the time series (*Extremum*, *Variance*, *Platform*), or they

temporarily change either the basic pattern positioning (*Mean*, *Pattern Shift*, *Trend*) or its shape (*Amplitude*, *Frequency*, *Pattern*) [48].



(a) Extremum (b) Variance (c) Platform

(d) Mean (e) Pattern Shift (f) Trend

(g) Amplitude (h) Frequency (i) Pattern

Figure 5.2: Examples of all 9 injected anomaly types.

While the synthetically generated datasets allow a deep and sophisticated analysis of the anomaly detection performance of DEAN-TS, the additional application on real datasets allows a more comprehensive assessment of its practical applicability. For this purpose, a selection of relevant univariate and multivariate time series from the NASA-MSL (Mars Science Laboratory) [21], NASA-SMAP (Soil Moisture Active Passive satellite) [21], and SMD (Server Machine Dataset) [21]. Dataset) [45] collections, which fully or partially match the selection datasets used for some of the competing related methods introduced in Chapter 5.1.2. A general summary of the main characteristics of the datasets considered can be found in Table 5.1.

Table 5.1: Datasets used for the benchmark evaluation with semi-supervised leaning.

Collection	**Origin**	**# Datasets**	**# Dimensions**	**Avg. Length**
GutenTAG [48]	Synthetic	193	1-20	10,000
NASA-MSL [21]	Real	27	1	2731
NASA-SMAP [21]	Real	54	1	8071
SMD [45]	Real	28	38	25300

5.1.2 Benchmark Algorithms

To evaluate the influence of parameterization options, the experiments are performed with several versions of DEAN-TS. Due to the large number of possible combinations, a default parameterization is chosen that seems reasonable which is then varied in one aspect for the different versions while keeping some baseline parameters constant. While a complete overview of the parameterization is given in Appendix B, the most relevant constant as well as all varying parameters can be found in Table 5.2 and Table 5.3. Due to the already high number of combinations and the expected higher added value, subsampling is deliberately used only in the unsupervised benchmark evaluation in Chapter 6.

The expected effect of these variations is as follows. Adding either the bias term or an additional hidden layer theoretically allows learning more complex relationships, but especially in the case of bias, it also increases the risk of degenerating submodels by finding effective but trivial solutions. Changing the activation function from ReLU to LeakyReLU [32] could possibly prevent the vanishing gradient problem sometimes observed in the original DEAN paper [25]. The reduction of *look_back_range* and $|lag\_indices|$ leads to a stronger focus on temporally close as opposed to more long-term relationships, and may potentially mitigate the problem of poorer performance for the first and last time steps addressed in Chapter 4.2.1. The exact modifications of the individual DEAN-TS versions and their subsequent naming are briefly outlined below. For a more complete listing and parameter naming consistent with the implementation, see Appendix B.

Table 5.2: Setting of the constant DEAN-TS parameters.

Ensemble Size	Combination	Feature Bagging	Subsampling
40	Thresh (threshold = 0)	True (range = [1,3])	False

Table 5.3: Default setting of the varying DEAN-TS parameters.

look_back_range	|lag_indices|	Bias	# Layers	Activation
[128,1024]	127	False	3	ReLU

DEAN-TS versions:

- **DEAN-TS:** Uses the exact default parameterization from Table 5.3.
- **DEAN-TS-Bag:** Lowers *look_back_range* to [64,512] and $|lag\_indices|$ to 63.
- **DEAN-TS-Bias:** Enables a bias term in the hidden layer.
- **DEAN-TS-Depth:** Increases the number of layers to 4.

- **DEAN-TS-Leaky:** Switches the activation function to LeakyReLU [32].

In order to fairly assess the strengths and weaknesses of the DEAN-TS methodology as well as the general difficulties of anomaly detection on the given datasets, the benchmarking was also performed with several related methods [44, 45, 34, 21]. The selection covers most of the method families introduced in Chapter 2.2.3 and is based on the best performance trade-off between anomaly detection quality and low failure rate (i.e. due to violation of time or resource constraints) observed in a comprehensive survey paper [43]. For all related methods, the respective open source implementation[1] of the TimeEval developers was used. The parameterization corresponds to the default values of the implementations and is detailed in Appendix B.

Related methods:

- **Hybrid KNN (Distance-based) [44]**: The *Hybrid KNN* method consists of two parts, a *Deep Autoencoder (DAE)* [19] and an ensemble of *k-nearest-neighbor (KNN)* [38] detectors. First, a nonlinear mapping of the training data to a more compact representation is learned unsupervised using a symmetric DAE. Then, multiple euclidean KNN detectors are created based on randomly selected subsamples of equal size with replacement. To determine the anomaly score of a data point, the estimated probability of its average observed k-th nearest neighbor distance is assessed.

- **OmniAnomaly(Reconstruction-based) [45]:** *OmniAnomaly* is a method based on stochastic recurrent neural networks (RNNs). It utilizes a complex network architecture combining gated recurrent units (GRUs) [8], a *Variational Autoencoder (VAE)* [24] and *planar normalizing flow (planar NF)* [39] to first transform each input with its consecutive previous observations into a latent representation and afterwards reconstruct it. The anomaly value is subsequently measured based on the reconstruction probability and compared to an automatically determined threshold.

- **LaserDBN (Encoding-based) [34]:** *LaserDBN* first uses two preprocessing steps to derive features. First, the range of values of the input data is discretized into a number of partitions. Afterwards, highly correlated features are grouped together by *hierarchical agglomerative clustering (HAC)*. For the detection of anomalies, dynamic bayesian networks (DBNs) as a generalization of hidden Markov models are used to represent the characteristics of normal behavior. The input are cluster features, which are extracted for each time step. Inputs for which the log likelihood with respect to the model is too low are

[1] https://github.com/HPI-Information-Systems/TimeEval-algorithms

- **Telemanom (Forecasting-based) [21]:** *Telemanom* generates a prediction for each time step and measures the degree of anomaly based on its deviation from the actual observed value. This is done by utilizing one independent LSTM based network per feature of a multivariate time series. For the clear differentiation between anomalous and normal behavior, a novel, non-parametric method for dynamic thresholding is used.

5.2 Anomaly Detection Performance (Synthetic Data)

Table 5.4 breaks down the average performance of all algorithms across all GutenTAG datasets based on each metric considered. Overall, Telemanom and Hybrid KNN performed best by a wide margin in terms of AUC-ROC. With respect to the two precision-recall based metrics, Hybrid KNN clearly performed best. Also noteworthy is the extremely high error rate of LaserDBN.

The different versions of DEAN-TS are comparatively competitive in the metrics AUC-PR and AUC-P_TR_T, but in terms of AUC-ROC they are rather behind the above mentioned related methods, but as well as in the other metrics they are clearly ahead of LaserDB and OmniAnomaly. Across all metrics, DEAN-TS-Bag is the best of all versions, with DEAN-TS coming in a close second. Comparable performance can be observed across the other 3 variants, with DEAN-TS-Depth falling slightly behind.

The following facts apply to the errors that occurred. On the synthetic GutenTAG datasets evaluated in this and the following subsections, one uncaused error per algorithm occurred due to an incorrectly formatted dataset. All remaining errors of the DEAN-TS versions as well as three of the four remaining errors of Telemanom are incomprehensible errors of TimeEval. All other errors of the corresponding methods refer to failures caused by the algorithm itself, but not by explicit violations of time or memory constraints.

Table 5.4: Anomaly detection performance for the GutenTAG datasets.

Algorithm	AUC-ROC	AUC-PR	AUC-P_TR_T	# Errors
DEAN-TS	$0.7042 + 0.0037$	$0.2985 + 0.0016$	$0.3393 + 0.0018$	1
DEAN-TS-Bag	$0.7113 + 0.0038$	$0.3525 + 0.0018$	$0.3808 + 0.0020$	1
DEAN-TS-Bias	$0.6784 + 0.0144$	$0.2908 + 0.0061$	$0.3251 + 0.0069$	4
DEAN-TS-Depth	$0.6742 + 0.0035$	$0.2751 + 0.0014$	$0.3009 + 0.0016$	1
DEAN-TS-Leaky	$0.6842 + 0.0108$	$0.2913 + 0.0046$	$0.3205 + 0.0051$	3
Hybrid KNN	$0.8339 + 0.0043$	$\mathbf{0.4683} + 0.0024$	$\mathbf{0.4419} + 0.0023$	1
LaserDBN	$0.5633 + 0.0877$	$0.1610 + 0.0251$	$0.1742 + 0.0272$	26
OmniAnomaly	$0.5682 + 0.0183$	$0.0808 + 0.0026$	$0.0780 + 0.0025$	6
Telemanom	$\mathbf{0.8538} + 0.0227$	$0.3622 + 0.0096$	$0.3743 + 0.0100$	5

The visualization of the achieved AUC-ROC scores across all GutenTAG datasets

given in Figure 5.3 reveals some additional insights. The different DEAN-TS versions show rather little variability in their performance spread. Especially for Hybrid KNN and Telemanom it is shown that their very good results exhibit little variance in the form of bad performance outliers. Additional visualizations for the in-depth analysis in the following subsections by different criteria are provided in Appendix C.

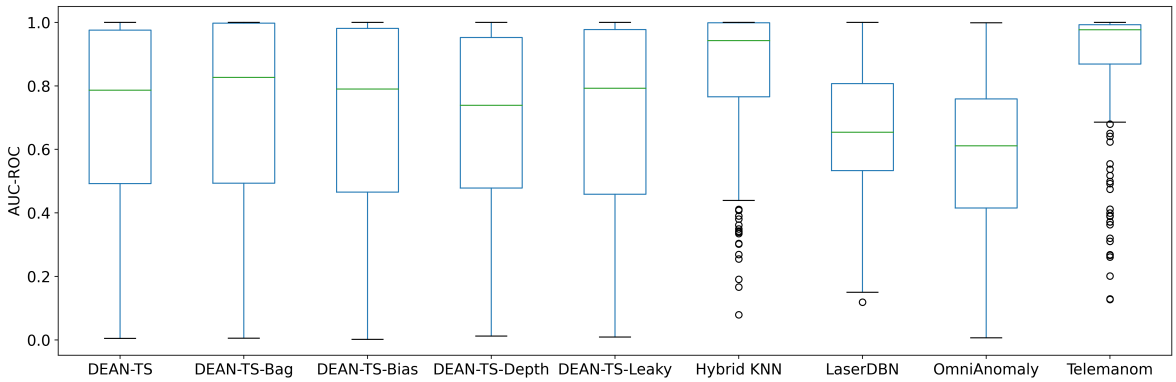


Figure 5.3: Box plot of AUC-ROC scores for the GutenTAG datasets.

5.2.1 Performance by Base Oscillation

When all synthetic data sets are grouped according to their underlying base oscillation, recognizable differences emerge both in the general difficulty of anomaly detection and in the varying influence of the base oscillation complexity on the individual algorithms. First, the maximum performance achieved for the periodic oscillations (see Table 5.5) is significantly higher than for the non-periodic ones (see Table 5.6).

Table 5.5: Anomaly detection performance by periodic base oscillation (AUC-ROC).

Algorithm	Sine	ECG
DEAN-TS	$0.9409 + 0.0000$	$0.7691 + 0.0000$
DEAN-TS-Bag	$\mathbf{0.9710} + 0.0000$	$0.7753 + 0.0000$
DEAN-TS-Bias	$0.8850 + 0.0603$	$0.7403 + 0.0000$
DEAN-TS-Depth	$0.8646 + 0.0000$	$0.7402 + 0.0000$
DEAN-TS-Leaky	$0.8897 + 0.0396$	$0.7767 + 0.0000$
Hybrid KNN	$0.9219 + 0.0000$	$0.8765 + 0.0000$
LaserDBN	$0.6247 + 0.0426$	$0.4204 + 0.1402$
OmniAnomaly	$0.6152 + 0.0133$	$0.5521 + 0.0263$
Telemanom	$0.9436 + 0.0205$	$\mathbf{0.9310} + 0.0000$

For the data sets with underlying sinusoidal oscillations, DEAN-TS-Bag performs best, for the rest it is either Telemanom or Hybrid KNN, with LaserDBN and OmniAnomaly

performing consistently poorly. While all DEAN-TS versions achieve very good results for an underlying sinusoidal oscillation and acceptable results for ECG-based oscillations, their performance for the non-periodic oscillations drops considerably and is always significantly behind those of Telemanom and Hybrid KNN. A definitive performance ranking between the DEAN-TS versions is difficult, since for example each of them except DEAN-TS-Bias performs best (often marginally) for at least one base oscillation.

Table 5.6: Anomaly detection performance by non-periodic base oscillation (AUC-ROC).

Algorithm	Poly	CBF	RW
DEAN-TS	0.6055 + 0.0000	0.5329 + 0.0144	0.5706 + 0.0000
DEAN-TS-Bag	0.6037 + 0.0000	0.5293 + 0.0143	0.5671 + 0.0000
DEAN-TS-Bias	0.5767 + 0.0000	0.5397 + 0.0145	0.5572 + 0.0000
DEAN-TS-Depth	0.5702 + 0.0000	0.5530 + 0.0150	0.5519 + 0.0000
DEAN-TS-Leaky	0.5481 + 0.0000	0.5126 + 0.0138	0.5938 + 0.0000
Hybrid KNN	**0.8847** + 0.0000	0.6698 + 0.0181	0.7928 + 0.0000
LaserDBN	0.5585 + 0.0598	0.5459 + 0.1023	0.6909 + 0.0691
OmniAnomaly	0.4754 + 0.0000	0.5621 + 0.0312	0.6172 + 0.0193
Telemanom	0.7846 + 0.0541	**0.6921** + 0.0385	**0.8743** + 0.0000

5.2.2 Performance by Anomaly Type

In the following analysis, only those datasets are considered in which a single anomaly type was injected, grouped by the corresponding anomaly type. This allows a better classification of the detection performance depending on the characteristics of an anomaly, so that clear advantages and disadvantages of individual methods become apparent depending on the prevalent anomaly type. Table 5.7 groups those anomalies that introduce anomalous artifacts, Table 5.8 those that temporarily change the base pattern positioning, and Table 5.9 those that change the pattern shape.

Table 5.7: Anomaly detection performance by anomaly type - part 1 (AUC-ROC).

Algorithm	Extremum	Variance	Platform
DEAN-TS	0.6138 + 0.0000	0.9108 + 0.0000	0.7838 + 0.0000
DEAN-TS-Bag	0.5657 + 0.0000	0.8690 + 0.0000	0.8035 + 0.0000
DEAN-TS-Bias	0.6015 + 0.0000	0.9084 + 0.0000	0.7729 + 0.0000
DEAN-TS-Depth	0.7129 + 0.0000	0.8839 + 0.0000	0.7124 + 0.0000
DEAN-TS-Leaky	0.6005 + 0.0000	0.8151 + 0.0000	0.7809 + 0.0000
Hybrid KNN	**0.9820** + 0.0000	0.9901 + 0.0000	0.6843 + 0.0000
LaserDBN	0.9680 + 0.0000	0.7586 + 0.0000	0.6366 + 0.0000
OmniAnomaly	0.6987 + 0.0000	0.6766 + 0.0000	0.4214 + 0.0000
Telemanom	0.7812 + 0.0000	**0.9922** + 0.0000	**0.9659** + 0.0000

Table 5.8: Anomaly detection performance by anomaly type - part 2 (AUC-ROC).

Algorithm	Mean	Pattern Shift	Trend
DEAN-TS	0.8529 + 0.0000	**0.9999** + 0.0000	0.5955 + 0.0000
DEAN-TS-Bag	0.8350 + 0.0000	**0.9999** + 0.0000	0.5294 + 0.0000
DEAN-TS-Bias	0.8504 + 0.0000	**0.9999** + 0.0000	0.5730 + 0.0000
DEAN-TS-Depth	0.7972 + 0.0000	**0.9999** + 0.0000	0.4936 + 0.0000
DEAN-TS-Leaky	0.7960 + 0.0000	**0.9999** + 0.0000	0.5277 + 0.0278
Hybrid KNN	0.8614 + 0.0000	0.9021 + 0.0000	**0.7703** + 0.0000
LaserDBN	0.6163 + 0.0000	0.5265 + 0.0000	0.6368 + 0.0000
OmniAnomaly	0.7943 + 0.0000	0.9008 + 0.0000	0.5075 + 0.0000
Telemanom	**0.9699** + 0.0000	0.9944 + 0.0000	0.6766 + 0.0751

Table 5.9: Anomaly detection performance by anomaly type - part 3 (AUC-ROC).

Algorithm	Amplitude	Frequency	Pattern
DEAN-TS	0.6969 + 0.0000	**0.9999** + 0.0000	0.9513 + 0.0000
DEAN-TS-Bag	0.6568 + 0.0000	**0.9999** + 0.0000	0.9606 + 0.0000
DEAN-TS-Bias	0.7014 + 0.0000	**0.9999** + 0.0000	0.9457 + 0.0000
DEAN-TS-Depth	0.6309 + 0.0000	**0.9999** + 0.0000	0.8398 + 0.0000
DEAN-TS-Leaky	0.6661 + 0.0000	**0.9999** + 0.0000	0.9489 + 0.0000
Hybrid KNN	0.8326 + 0.0000	**0.9999** + 0.0000	0.8952 + 0.0000
LaserDBN	0.5655 + 0.0000	0.6307 + 0.0000	0.4799 + 0.0000
OmniAnomaly	0.7410 + 0.0000	0.5722 + 0.0000	0.6030 + 0.0000
Telemanom	**0.9673** + 0.0000	0.9962 + 0.0000	**0.9646** + 0.0000

In general, the best performance is achieved with the pattern shift and frequency anomaly types. Here, all DEAN-TS versions achieve an almost perfect AUC-ROC score, which is only matched by Hybrid KNN and Telemanom for frequency anomalies and by Telemanom for pattern shift anomalies. It should be noted that both pattern shift and frequency anomalies were only injected into the datasets with periodic base oscillations.

For mean, variance, and especially pattern anomalies, several of the DEAN-TS versions achieve good or very good, and for platform anomalies at least acceptable results, but not for extremum, amplitude, and trend anomalies. In general, the latter seem to be the most difficult to detect, as neither Telemanom nor Hybrid KNN achieve a score > 0.78. The performance between these two varies considerably, with Telemanom performing much better on extremum anomalies whereas Hybrid KNN performs much better on platform anomalies. Among the DEAN-TS versions, DEAN-TS-Depth is the only one that achieves an acceptable but still not competitive result for extremum anomalies, but at the same time it performs the worst of all versions for amplitude, trend, platform, and pattern anomalies. Again, a definitive ranking is difficult, as every version except DEAN-TS-Leaky performs best on at least one anomaly type.

5.2.3 Performance by Dimensionality and Contamination

We now consider the influence of the simultaneously considered variables, the dimensionality (see Table 5.10), as well as the number of anomalies in the data set, the contamination (see Table 5.11). It is noteworthy that the AUC-ROC for all related methods decreases by about 0.1 for multivariate time series compared to their univariate counterparts. Furthermore, it seems that a disproportionate number of LaserDBN experiments fail in the multivariate case.

Table 5.10: Anomaly detection performance by dimensionality (AUC-ROC).

Algorithm	Univariate	Multivariate
DEAN-TS	$0.7061 + 0.0000$	$0.6916 + 0.0288$
DEAN-TS-Bag	$0.7171 + 0.0000$	$0.6730 + 0.0280$
DEAN-TS-Bias	$0.6816 + 0.0124$	$0.6571 + 0.0274$
DEAN-TS-Depth	$0.6765 + 0.0000$	$0.6584 + 0.0274$
DEAN-TS-Leaky	$0.6884 + 0.0083$	$0.6559 + 0.0273$
Hybrid KNN	$0.8465 + 0.0000$	$0.7491 + 0.0312$
LaserDBN	$0.6227 + 0.0353$	$0.1639 + 0.3484$
OmniAnomaly	$0.5801 + 0.0142$	$0.4885 + 0.0425$
Telemanom	$\mathbf{0.8642} + 0.0211$	$\mathbf{0.7841} + 0.0327$

For the DEAN-TS versions, the absolute performance loss due to multiple dimensions is significantly lower than for the other methods, although the AUC-ROC for univariate time series is also disproportionately lower than for Hybrid KNN or the generally best performing Telemanom in the first place. In both the univariate and the multivariate case, DEAN-TS and DEAN-TS-Bag perform best of all DEAN-TS versions, with an AUC-ROC score difference of only about 0.01.

Table 5.11: Anomaly detection performance by number of anomalies (AUC-ROC).

Algorithm	1 Anomaly	1 - 5 Anomalies	$>$ 5 Anomalies
DEAN-TS	$0.7021 + 0.0000$	$0.7566 + 0.0223$	$0.6535 + 0.0000$
DEAN-TS-Bag	$0.7029 + 0.0000$	$0.7743 + 0.0227$	$0.6748 + 0.0000$
DEAN-TS-Bias	$0.6817 + 0.0054$	$0.7476 + 0.0219$	$0.5872 + 0.0404$
DEAN-TS-Depth	$0.6536 + 0.0000$	$0.7578 + 0.0223$	$0.6640 + 0.0000$
DEAN-TS-Leaky	$0.6747 + 0.0108$	$0.7583 + 0.0223$	$0.6394 + 0.0000$
Hybrid KNN	$0.8490 + 0.0000$	$0.8103 + 0.0238$	$0.7984 + 0.0000$
LaserDBN	$0.5933 + 0.0795$	$0.4778 + 0.0988$	$0.5367 + 0.1032$
OmniAnomaly	$0.5560 + 0.0089$	$0.6019 + 0.0564$	$0.5806 + 0.0194$
Telemanom	$\mathbf{0.8596} + 0.0208$	$\mathbf{0.8181} + 0.0496$	$\mathbf{0.8705} + 0.0000$

Notably, for data sets with one to five anomalies, all DEAN-TS versions perform better than those with only one or more than five anomalies, with DEAN-TS-Bag coming close

to the performance of Telemanom and Hybrid KNN. This phenomenon is otherwise only seen for OmniAnomaly, with the other related methods otherwise dropping in performance as the contamination increases from one to multiple anomalies. Also, all versions except DEAN-TS-Depth show their worst performance for datasets with more than 5 anomalies.

5.3 Anomaly Detection Performance (Real Data)

We now examine the performance of DEAN-TS and related methods on several real-world datasets. Since the SMD datasets consider numerous time steps and variables, we have limited ourselves to the DEAN-TS-Bag version and the related methods Telemanom and OmniAnomaly, because they behaved comparatively well in this environment in terms of erroneously terminated experiments and runtime characteristics. A breakdown of the individual errors is provided in the subsections of the respective data set collections. For the more manageable NASA dataset collection NASA-MSL and NASA-SMAP, we still applied each of the previously used benchmark algorithms.

Since only the aforementioned selection of the algorithms were applied to all real data sets, only those are included in the overall summary Table 5.12, as well as in Figure 5.4. The composition of all errors listed here is broken down further in the specific subsections.

Table 5.12: Anomaly detection performance for all real datasets.

Algorithm	AUC-ROC	AUC-PR	AUC-P_TR_T	# Errors
DEAN-TS-Bag	$0.7708 + 0.0294$	$0.4920 + 0.0187$	$0.4499 + 0.0171$	4
OmniAnomaly	$0.7278 + 0.0206$	$0.3649 + 0.0103$	$0.3061 + 0.0086$	3
Telemanom	$\mathbf{0.8239} + 0.0077$	$\mathbf{0.5401} + 0.0050$	$\mathbf{0.4996} + 0.0046$	1

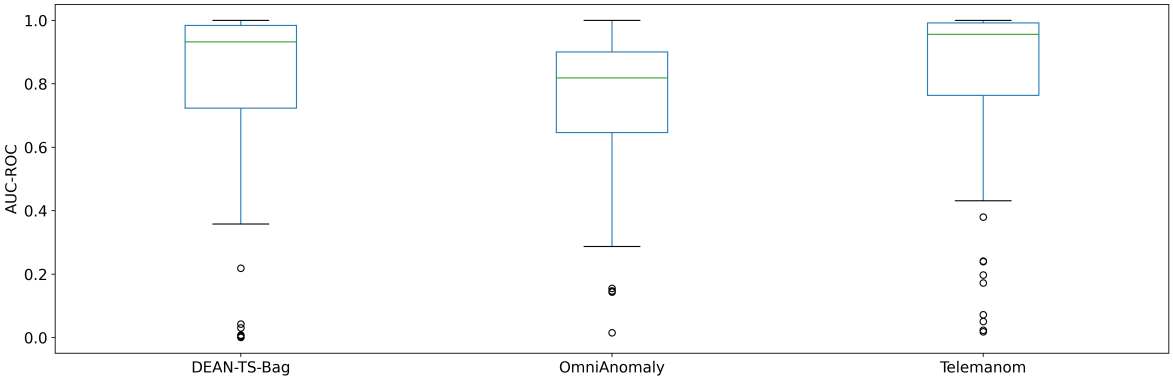


Figure 5.4: Box plot of AUC-ROC scores for all real datasets.

While Telemanom is again the best overall performer, we can also see that DEAN-TS-Bag is quite competitive. OmniAnomaly also drops off less than for the GutenTAG

datasets, at least in terms of AUC-ROC, but still lags behind DEAN-TS-Bag and Telemanom in all other metrics. The box plots of the AUC values in Figure 5.4 show a similar picture, with the median values of 0.9561 for Telemanom, 0.9257 for DEAN-TS-Bag, and 0.8143 for OmniAnomaly highlighting how some outliers drag down the generally very good performance. For the precision-recall based metrics, both DEAN-TS-Bag and Telemanom perform significantly better than on the GutenTAG datasets.

5.3.1 NASA-MSL and NASA-SMAP

Both NASA dataset collections are assemblages of univariate time series datasets. While the NASA-SMAP time series typically contain more time steps, the anomalies occurring in this collection have been described as easier to detect [21]. Also, the NASA-MSL time series are said to represent a wider range of different anomaly types [21], so weaker performance would be expected here.

As we can see in the Table 5.13 and 5.14, Telemanom, which was developed by NASA employees and initially tested in their original work on these two dataset collections, again performs best on all metrics. Consistent with the previously formulated expectation, it achieves higher absolute values on all metrics for the NASA SMAP collection. In terms of AUC-ROC, OmniAnomaly outperforms Hybrid KNN for both collections, but this is reversed for the precision-recall based metrics. LaserDBN is again consistently the worst performer and also has the highest error rate.

Table 5.13: Anomaly detection performance for the NASA-MSL datasets.

Algorithm	AUC-ROC	AUC-PR	AUC-P_TR_T	# Errors
DEAN-TS	$0.5910 + 0.1028$	$0.3436 + 0.0597$	$0.3582 + 0.0623$	4
DEAN-TS-Bag	$0.7506 + 0.0288$	$0.4748 + 0.0182$	$0.4699 + 0.0181$	1
DEAN-TS-Bias	$0.5853 + 0.1018$	$0.3296 + 0.0574$	$0.3507 + 0.0610$	4
DEAN-TS-Depth	$0.5740 + 0.0998$	$0.3497 + 0.0608$	$0.3872 + 0.0673$	4
DEAN-TS-Leaky	$0.5896 + 0.1025$	$0.3468 + 0.0603$	$0.3743 + 0.0651$	4
Hybrid KNN	$0.6836 + 0.0000$	$0.4173 + 0.0000$	$0.3942 + 0.0000$	0
LaserDBN	$0.3402 + 0.1191$	$0.1212 + 0.0424$	$0.1002 + 0.0351$	7
OmniAnomaly	$0.7174 + 0.0276$	$0.3448 + 0.0133$	$0.3437 + 0.0132$	1
Telemanom	$\mathbf{0.8101} + 0.0000$	$\mathbf{0.5022} + 0.0000$	$\mathbf{0.4851} + 0.0000$	0

For the NASA-MSL datasets, DEAN-TS-Bag, in contrast to the other DEAN-TS versions, shows an approximately comparable performance to Telemanom, especially with respect to the precision-recall metrics, and is the second best overall. All other DEAN-TS versions show a rather mixed performance, which does not vary too much among each other, whereby the higher number of erroneous runs also plays a role here. For the NASA-SMAP datasets, DEAN-TS-BAG performs somewhat worse than the other versions, at

Table 5.14: Anomaly detection performance for the NASA-SMAP datasets.

Algorithm	AUC-ROC	AUC-PR	AUC-P_TR_T	# Errors
DEAN-TS	$0.7554 + 0.0771$	$0.5339 + 0.0545$	$0.5054 + 0.0516$	5
DEAN-TS-Bag	$0.7285 + 0.0429$	$0.5297 + 0.0312$	$0.5244 + 0.0308$	3
DEAN-TS-Bias	$0.6958 + 0.1210$	$0.4957 + 0.0862$	$0.4760 + 0.0827$	8
DEAN-TS-Depth	$0.7507 + 0.1118$	$0.5103 + 0.0760$	$0.4803 + 0.0715$	7
DEAN-TS-Leaky	$0.7392 + 0.0924$	$0.5211 + 0.0652$	$0.4899 + 0.0612$	6
Hybrid KNN	$0.6444 + 0.0000$	$0.4592 + 0.0000$	$0.4399 + 0.0000$	0
LaserDBN	$0.2792 + 0.0886$	$0.1252 + 0.0397$	$0.1333 + 0.0422$	13
OmniAnomaly	$0.6871 + 0.0130$	$0.3855 + 0.0073$	$0.3506 + 0.0066$	1
Telemanom	$\mathbf{0.8347} + 0.0000$	$\mathbf{0.6545} + 0.0000$	$\mathbf{0.6606} + 0.0000$	0

least with respect to AUC-ROC, but still shows the fewest erroneous runs.

Analyzing the box plots in Figure 5.5 and Figure 5.6, it is interesting to see that the median performance of all DEAN-TS versions, especially DEAN-TS-Bag, for the NASA-SMAP datasets is significantly higher than their mean, but especially for DEAN-TS some comparatively bad performances shift the mean strongly downwards. In general, for both box plots, the median of all DEAN-TS versions is again significantly higher than their mean, for the same reason.

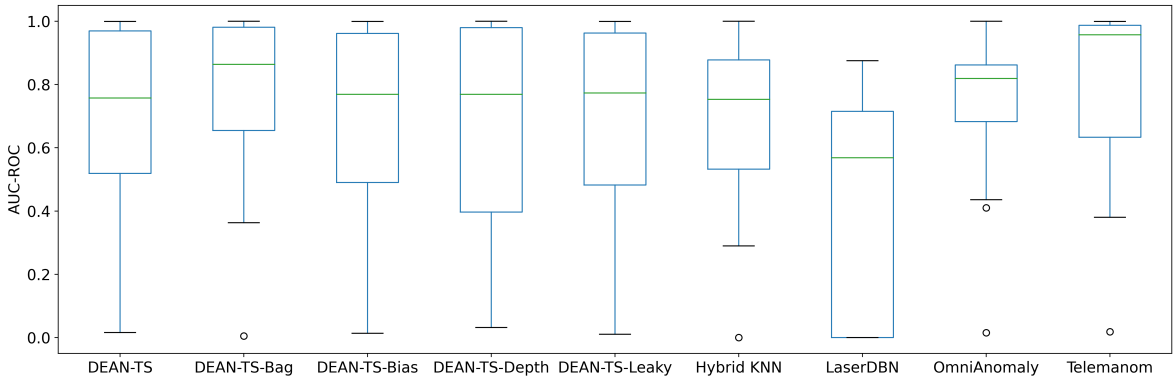


Figure 5.5: Box plot of AUC-ROC scores for the NASA-MSL datasets.

All errors of the DEAN-TS versions on the NASA-MSL datasets are caused by a now fixed programming error when training on a comparatively small sample size. The multiple zero scores resulting from this error hurt the average performance of the DEAN-TS versions noticeably, especially since competitors such as Telemanom performed disproportionately well on these datasets. The same error also affects DEAN-TS-Bag once on the NASA-SMAP datasets, as well as each of the other versions five times. All remaining errors of the DEAN-TS versions are again on the part of TimeEval. The errors of LaserDBN and

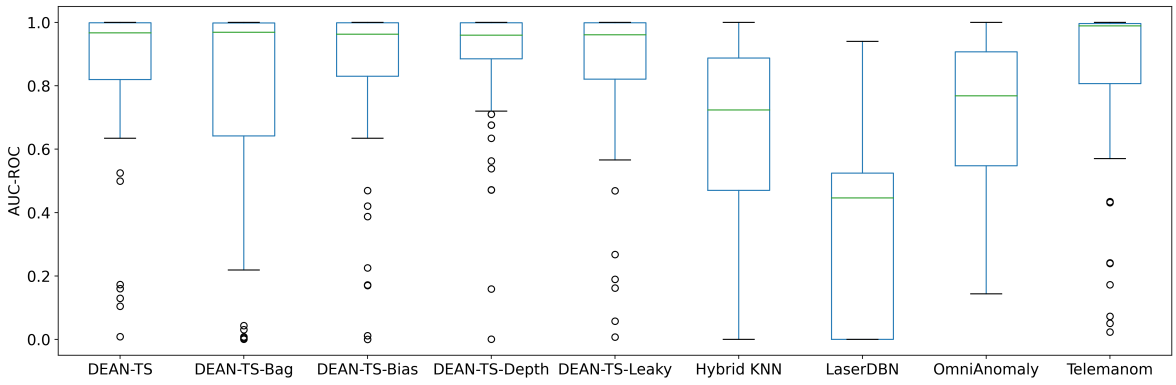


Figure 5.6: Box plot of AUC-ROC scores for the NASA-SMAP datasets.

OmniAnomaly seem to be entirely caused by the respective algorithms.

5.3.2 SMD

The SMD dataset collection consists of time series with a higher complexity than any of the previously considered ones, both in terms of length and dimensionality. As a result, the number of common errors increased (Hybrid KNN and LaserDBN) and the runtime constraints were frequently violated (DEAN-TS versions) despite an increase to 30 minutes. Accordingly, the following experiments were performed using only the more runtime efficient DEAN-TS-Bag version (see Chapter 5.4) as well as OmniAnomaly and Telemanom. The two remaining errors of the corresponding methods were uncaused by the algorithms and on the part of TimeEval.

Table 5.15: Anomaly detection performance for the SMD datasets.

Algorithm	AUC-ROC	AUC-PR	AUC-P_TR_T	# Errors
DEAN-TS-Bag	**0.8719** + 0.0000	**0.4358** + 0.0000	**0.2868** + 0.0000	0
OmniAnomaly	0.8165 + 0.0303	0.3445 + 0.0128	0.1839 + 0.0068	1
Telemanom	0.8166 + 0.0302	0.3560 + 0.0132	0.2033 + 0.0075	1

As can be seen in Table 5.15, DEAN-TS-Bag remarkably achieves the best average performance in anomaly detection on these highly complex time series. This particularly holds true for all considered metrics with a significant lead. As with the other dataset collections, the AUC-ROC median is still slightly higher than the corresponding mean, but the box plots in Figure 5.7 show a smaller lead over Telemanom and a similar variance in performance for these two methods.

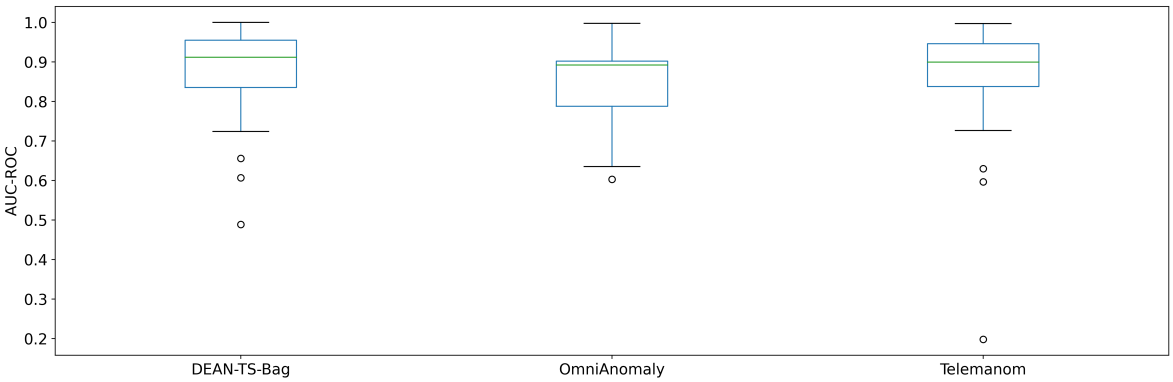


Figure 5.7: Box plot of AUC-ROC scores for the SMD datasets.

5.4 Runtime Efficiency

Besides the raw anomaly detection performance, secondary requirements such as runtime efficiency can be important for the practical applicability of a method. This is true for both the training time, which can be significant for certain deep learning methods, and the subsequent prediction time, which can become an important factor in time-critical applications. Figure 5.8 compares the runtime of all previously evaluated algorithms across all GutenTAG datasets, split into training and prediction time in ascending order of their total runtime requirements. A logarithmic scaling is applied for better visualization of the rather runtime efficient LaserDBN and Hybrid KNN. In order to differentiate between quite similar runtime performances, the accompanying Table 5.16 details the exact average runtime component values, further split into univariate and multivariate time series.

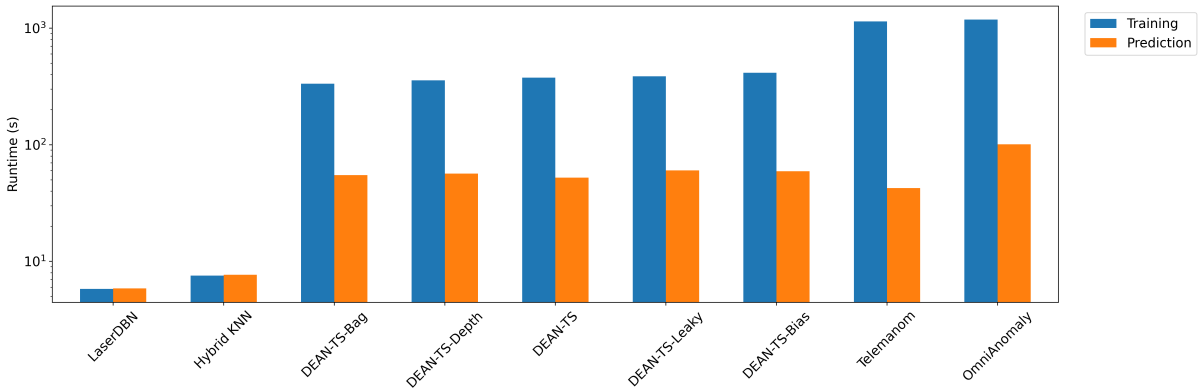


Figure 5.8: Bar chart of the average runtime for the GutenTAG datasets in seconds with log scale.

Both of the aforementioned methods, LaserDBN and Hybrid KNN, show a significantly lower total runtime than the other methods. Moreover, they are the only cases where

training time is not the dominant factor. While the logarithmic scaling makes it less visible in the box plots, the DEAN-TS-Bag method was the most runtime efficient DEAN-TS version, as expected due to the smaller bag size, while DEAN-TS-Bias took the longest on average for both training and prediction. While Telemanom, like OmniAnomaly, is generally slightly faster than the DEAN-TS versions for prediction, it takes 2-3 times as long for training in both the univariate and multivariate cases. Notably, the runtime does not seem to increase significantly with increasing dimensionality for any of the methods from related work, and in some cases it even decreases.

Table 5.16: Average runtime of training and prediction steps for the GutenTAG datasets.

Algorithm	**Univariate**		**Multivariate**	
	Training	**Prediction**	**Training**	**Prediction**
DEAN-TS	376.72s	52.21s	486.16s	77.82s
DEAN-TS-Bag	334.17s	54.84s	390.11s	57.83s
DEAN-TS-Bias	414.20s	59.17s	580.94s	92.48s
DEAN-TS-Depth	356.07s	56.61s	436.63s	88.79s
DEAN-TS-Leaky	385.77s	60.22s	462.13s	87.13s
Hybrid KNN	7.55s	7.68s	7.92s	7.59s
LaserDBN	5.80s	5.87s	2.10s	2.25s
OmniAnomaly	1183.80s	100.84s	1115.67s	92.82s
Telemanom	1142.80s	42.49s	1163.94s	43.90s

When evaluating the runtime properties of DEAN-TS, it should be noted that the extreme parallelization potential has not been exploited in the current implementation. In contrast to some other methods, whose procedure partly insists on a necessarily sequential execution, e.g. OmniAnomaly [45], in the case of DEAN-TS all runtime-intensive steps can theoretically be parallelized very well. Namely, the preprocessing, training and scoring of new instances by individual submodels can be performed completely independent of the other submodels. Thus, an improvement by a factor of the ensemble size should be easily achievable by implementing explicit parallelization in combination with an execution platform with suitable hardware capacity.

5.5 Concluding Assessment

In Table 5.17, the algorithms analyzed are ranked in terms of the average performance achieved for each metric and for all the dataset collections considered. In the ranking with respect to AUC-ROC, Telemanom's extremely good performance stands out again, which is also observed with a slight decrease for the other metrics. This could be due to the low number of inter-metric anomalies in the datasets, which suits Telemanom's approach of training a single model for each variable. Hybrid KNN performs very well on

the GutenTAG datasets for all metrics, but drops off a bit otherwise (although it was not considered for the SMD collection), but shows extremely good runtime efficiency, which could possibly justify the performance loss depending on the use case. OmniAnomaly never makes it to one of the first two places, and LaserDBN never even makes it to one of the first three. While these and the following evaluations give some indication, the anomaly detection quality and especially the runtime efficiency of the methods are strongly influenced by the respective implementations and parameterizations (which are listed in Appendix B). Therefore, the obtained results should not be taken as a final quality assessment of a method [26].

Table 5.17: Average performance ranking by dataset collection for each metric.

Metric	Collection	Best	2nd Best	3rd Best
AUC-ROC	GutenTAG	Telemanom	Hybrid-KNN	DEAN-TS-Bag
	NASA-MSL	Telemanom	DEAN-TS-BAG	OmniAnomaly
	NASA-SMAP	Telemanom	DEAN-TS	DEAN-TS-Depth
	SMD	DEAN-TS-BAG	Telemanom	OmniAnomaly
AUC-PR	GutenTAG	Hybrid KNN	Telemanom	DEAN-TS-Bag
	NASA-MSL	Telemanom	DEAN-TS-BAG	Hybrid KNN
	NASA-SMAP	Telemanom	DEAN-TS	DEAN-TS-Bag
	SMD	DEAN-TS-BAG	Telemanom	OmniAnomaly
AUC-P_TR_T	GutenTAG	Hybrid KNN	DEAN-TS-Bag	Telemanom
	NASA-MSL	Telemanom	DEAN-TS-BAG	Hybrid KNN
	NASA-SMAP	Telemanom	DEAN-TS-Bag	DEAN-TS
	SMD	DEAN-TS-BAG	Telemanom	OmniAnomaly

Among the DEAN-TS versions, DEAN-TS-Bag generally performs the best, even ignoring the fact that it was the only version used for the SMD collection. Notably, the lower *look_back_range* chosen here, as well as the selection of fewer lag indices, seems to have a measurable positive effect on the AUC-P_TR_T metric in particular. Furthermore, its performance on the NASA-MSL and especially the SMD datasets was generally very good. While sporadically some of the other DEAN-TS versions performed better when grouping the GutenTAG datasets by certain features, e.g. the additional layer of DEAN-TS-Depth for extremum anomaly detection, mostly only the standard DEAN-TS version performed comparably well. In general, for all versions of DEAN-TS, the more robust median is often much higher than the mean, which can be explained by some comparatively very poor detection performances. In terms of runtime, DEAN-TS-Depth and especially DEAN-TS-Bag were slightly better than the other versions, but overall quite similar.

In summary, it seems that DEAN-TS is comparatively insensitive to hyperparameters. Including bias or an additional layer, as well as changing the activation function to Leaky-ReLU seems to be rather detrimental, except in rare cases. Due to the tendency of better

performance and slightly better runtime, it could be experimented to further reduce the window size and the number of lag indices compared to DEAN-TS-Bag, relative to the length of the time series.

Generally, it can also be observed that several of the trends identified in the original survey paper using TimeEval [43] reappear. A general finding is that no algorithm performs consistently best over all possible time series characteristics, and that especially for more complex anomaly detection conditions there is still room for improvement for all methods. While frequency and pattern shift anomalies, rather than extremum anomalies, were detected most reliably, trend anomalies still seem to be the most difficult to detect. In addition, anomaly detection was usually more successful on periodic base oscillations than on non-periodic ones.

6 Benchmark Evaluation (Unsupervised)

This chapter uses the same benchmark evaluation approach as previously to evaluate the anomaly detection performance and runtime efficiency of DEAN-TS, but this time without prior knowledge of normal behavior, i.e., in an unsupervised learning environment where anomalies are present in the training data. Again, several DEAN-TS versions are experimentally investigated, this time using different subsampling approaches. A direct visual comparison with the semi-supervised performance has been included in Appendix C.

6.1 Environment and Setup

The basic setup of the experiments is the same as described in Chapter 5.1. Specifically, the experiments were run using the TimeEval benchmark tool [48] on the same hardware with the same restrictions regarding CPU cores, memory usage and a maximum runtime of 20 minutes. The error handling for presenting the results is also done in the same way.

6.1.1 Benchmark Datasets

In the following, we restrict ourselves to the test versions of the time series datasets introduced in Chapter 5.1.1, which were synthetically generated with GutenTAG. Since the same data is evaluated in the testing as in the supervised learning counterpart, while the training has now taken place with the inclusion of anomalies, the direct influence of this factor can be quantified. A visualization of this is provided in Appendix C. For the sake of completeness, Table 6.1 once again lists the main characteristics of the datasets.

Table 6.1: Datasets used for the benchmark evaluation with unsupervised leaning.

Collection	Origin	# Datasets	# Dimensions	Avg. Length
GutenTAG [48]	Synthetic	193	1-20	10,000

6.1.2 Benchmark Algorithms

Since DEAN-TS-Bag performed best among the evaluated DEAN-TS versions in the semi-supervised benchmarking, we use its parameterization as a starting point. In particular, we always choose an ensemble size of 40 as well as the threshold method(threshold=0) for combination and feature bagging, considering 1-3 variables of multivariate time series per submodel. With DEAN-TS-Bag as a baseline, all subsequent versions use the reduced values for $|look\_back|$ and $|lag\_indices|$, omit a bias term and an additional layer, and stick to the ReLU activation function. Motivated by the hoped-for advantages of structured subsampling mentioned in Chapter 4.2.2, we now compare training and testing on the

same complete time series used with structured and random subsampling. Again, for a more complete listing and parameter naming consistent with the implementation, see Appendix B.

DEAN-TS versions:

- **DEAN-TS-Bag:** Same parametrization as denoted in Chapter 5.1.2
- **DEAN-TS-Bag-SS:** Combines DEAN-TS-Bag with structured subsampling using $r = [0.1, 0.25]$ and $m = [2, 5]$. This results in 2 trained submodels for each tenth of the time series and 5 for each fourth, non-overlapping in each case, leading to a ensemble size of 40 as well.
- **DEAN-TS-Bag-RS:** Combines DEAN-TS-Bag with random subsampling using $rs\_range = [1024, 8192]$, so that each of the 40 submodels is trained on a random subsample with a sample size between 1024 and 8192.

In order to better assess the performance of DEAN-TS in unsupervised learning, not only compared to its application in supervised learning, but also in general, the following experiments are also conducted with some alternative approaches. Although these methods are comparatively simplistic and, in particular, do not use deep learning, they were ranked 2nd and 4th in the original study in terms of average performance over all datasets when it comes to unsupervised methods suitable for multivariate time series [43]. Furthermore, unlike the other top four methods, their application never violated the imposed memory or time constraints [43].

Related methods:

- **Extended Isolation Forest (Tree-based) [17]:** As the name suggests, the *Extended Isolation Forest (EIF)* is an extension of the usual Isolation Forest method. Here it is allowed to partition the data using not necessarily axes-parallel hyperplanes with random slope instead of just straight lines.
- **KNN (Distance-based) [38]:** This is a standard KNN approach in which a data point is evaluated based on its euclidean distance to the k-th nearest neighbor across all data points in the time series.

6.2 Anomaly Detection Performance (Synthetic Data)

Summarizing the average anomaly detection performance across all GutenTAG datasets in Table 6.2, it is apparent that the DEAN-TS versions generally outperform EIF and

KNN for all metrics. The omission of subsampling in DEAN-TS-Bag seems to have had a positive effect on the AUC-ROC performance, while there are no major differences with respect to the precision-recall metrics. Both subsampling variants generally achieved similar results for this high-level aggregation. Regarding the errors that occurred, one per algorithm again resulted from an incorrectly formatted data set. The remaining errors in the DEAN-TS versions are all due to errors caused by TimeEval.

Table 6.2: Anomaly detection performance for the GutenTAG datasets (unsupervised).

Algorithm	AUC-ROC	AUC-PR	AUC-P_TR_T	# Errors
DEAN-TS-Bag	**0.6715** + 0.0142	0.3065 + 0.0064	**0.3386** + 0.0072	4
DEAN-TS-Bag-RS	0.6274 + 0.0066	**0.3118** + 0.0033	0.3226 + 0.0034	2
DEAN-TS-Bag-SS	0.6423 + 0.0067	0.3037 + 0.0032	0.3139 + 0.0033	2
EIF	0.5987 + 0.0031	0.1204 + 0.0006	0.1871 + 0.0010	1
KNN	0.5943 + 0.0031	0.1549 + 0.0008	0.1725 + 0.0009	1

The visualization of the AUC-ROC score distributions over the individual data sets by the box plots in Figure 6.1 shows no substantial differences between the median values and the mean values listed in Table 6.2. At the same time, the performance fluctuations observed for all DEAN-TS versions are comparatively high, especially when subsampling is used.

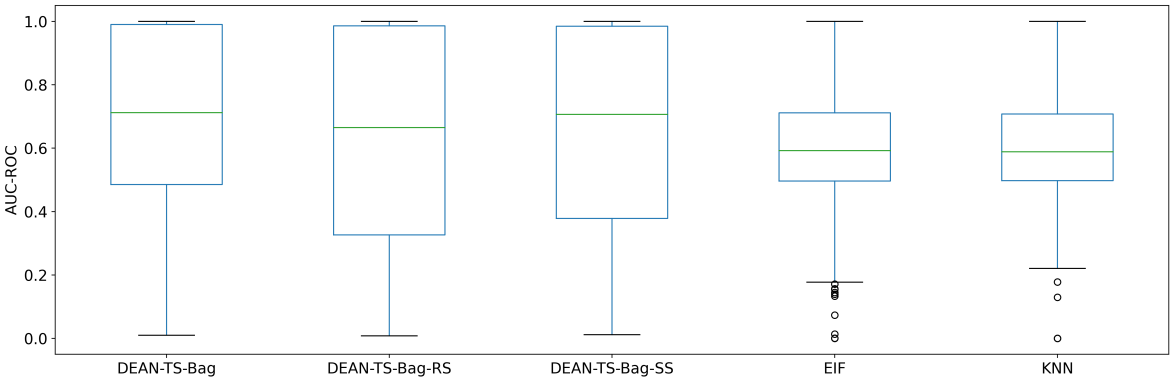


Figure 6.1: Box plot of AUC-ROC scores for the GutenTAG datasets (unsupervised).

6.2.1 Performance by Base Oscillation

As in the semi-supervised setting, the AUC-ROC values for the periodic base oscillations (see Table 6.3) Sine and ECG are by far higher than for the non-periodic ones (see Table 6.4). Despite the anomalies now present in the training data, the obtained result was even slightly better for the sinusoidal oscillations. For the other base oscillations,

there was a larger drop in performance, so that EIF and KNN consistently performed significantly better than all DEAN-TS versions. Subsampling seems to have been uniformly detrimental in this form of result aggregation, with structured subsampling still yielding better results than random subsampling, except for the CBF oscillation.

Table 6.3: Anomaly detection performance by periodic base oscillation (AUC-ROC).

Algorithm	**Sine**	**ECG**
DEAN-TS-Bag	**0.9740** + 0.0000	**0.7517** + 0.0175
DEAN-TS-Bag-RS	0.9062 + 0.0197	0.7096 + 0.0000
DEAN-TS-Bag-SS	0.9308 + 0.0000	0.7369 + 0.0000
EIF	0.5675 + 0.0000	0.5381 + 0.0000
KNN	0.6016 + 0.0000	0.5722 + 0.0000

Table 6.4: Anomaly detection performance by non-periodic base oscillation (AUC-ROC).

Algorithm	**Poly**	**CBF**	**RW**
DEAN-TS-Bag	0.5688 + 0.0189	0.4559 + 0.0254	0.4787 + 0.0000
DEAN-TS-Bag-RS	0.4840 + 0.0000	0.4786 + 0.0130	0.4268 + 0.0000
DEAN-TS-Bag-SS	0.5102 + 0.0000	0.4611 + 0.0256	0.4382 + 0.0000
EIF	**0.6925** + 0.0000	0.5782 + 0.0156	**0.6595** + 0.0000
KNN	0.6499 + 0.0000	**0.5829** + 0.0157	0.5742 + 0.0000

6.2.2 Performance by Anomaly Type

When grouping by anomaly type, we see very similar structures as in the semi-supervised evaluation. Again, this done with Table 6.5 considering those anomalies that introduce anomalous artifacts, Table 6.6 those that temporarily change the base pattern positioning, and Table 6.7 those that change the pattern shape.

For the pattern shift and frequency anomalies, all DEAN-TS versions once again performed extremely well, although for DEAN-TS-Bag the uncaused error message shifts the score significantly, at least at first glance. For the mean, variance and pattern anomalies, subsampling seems to hurt the otherwise rather good performance. The same is true for amplitude, trend and extremum anomalies, where DEAN-TS-Bag itself does not show a good baseline performance. On the other hand, subsampling seems to have helped the platform anomaly detection. The corresponding methods almost consistently perform worse than the DEAN-TS versions, except for trend anomalies, where they achieve comparable results to DEAN-TS-Bag, and a near perfect detection performance for extremum anomalies in both cases.

Table 6.5: Anomaly detection performance by anomaly type - part 1 (AUC-ROC).

Algorithm	Extremum	Variance	Platform
DEAN-TS-Bag	0.6859 + 0.0000	**0.8875** + 0.0000	0.7068 + 0.0000
DEAN-TS-Bag-RS	0.5352 + 0.0000	0.7369 + 0.0000	**0.7902** + 0.0000
DEAN-TS-Bag-SS	0.5971 + 0.0000	0.7885 + 0.0000	0.7616 + 0.0000
EIF	0.9978 + 0.0000	0.6573 + 0.0000	0.3835 + 0.0000
KNN	**0.9997** + 0.0000	0.7169 + 0.0000	0.3648 + 0.0000

Table 6.6: Anomaly detection performance by anomaly type - part 2 (AUC-ROC).

Algorithm	Mean	Pattern Shift	Trend
DEAN-TS-Bag	**0.8597** + 0.0000	0.4997 + 0.4997	0.5644 + 0.0000
DEAN-TS-Bag-RS	0.7465 + 0.0000	0.9998 + 0.0000	0.3808 + 0.0200
DEAN-TS-Bag-SS	0.7239 + 0.0000	**0.9999** + 0.0000	0.4372 + 0.0000
EIF	0.6591 + 0.0000	0.5056 + 0.0000	0.5505 + 0.0000
KNN	0.6373 + 0.0000	0.5273 + 0.0000	**0.5645** + 0.0000

Table 6.7: Anomaly detection performance by anomaly type - part 3 (AUC-ROC).

Algorithm	Amplitude	Frequency	Pattern
DEAN-TS-Bag	**0.6458** + 0.0000	0.9996 + 0.0000	**0.8871** + 0.0000
DEAN-TS-Bag-RS	0.6212 + 0.0000	**0.9999** + 0.0000	0.7784 + 0.0000
DEAN-TS-Bag-SS	0.5611 + 0.0000	**0.9999** + 0.0000	0.7530 + 0.0000
EIF	0.5337 + 0.0000	0.4880 + 0.0000	0.5102 + 0.0000
KNN	0.5608 + 0.0000	0.4890 + 0.0000	0.4672 + 0.0000

6.2.3 Performance by Dimensionality and Contamination

For multivariate time series, as can be seen in Table 6.8, there is a clear drop in performance for DEAN-TS-Bag compared to the univariate time series. However, this is almost exclusively due to errors not induced by the algorithm. Notably, the performance of the subsampling variants increases for multiple dimensions in both cases, as it does for the related methods. In the case of DEAN-TS-Bag-SS and EIF this is even significant, so that the latter two candidates show an acceptable, comparatively good performance. The additional dimensions to be considered do not seem to make a relevant difference in the case of KNN.

In addition to improving runtime efficiency, Chapter 4.2.2 identifies the theoretical mitigation of the influence of anomalies in unsupervised training as a major opportunity of subsampling. This is justified by the fact that now not all submodels need to include these anomalies when learning normal structures. In fact, a trend towards this can be seen in Table 6.8. For more than one anomaly, at least random subsampling performs better than DEAN-TS-Bag. The use of structural subsampling, on the other hand, shows

no such effect. The related methods lag behind all DEAN-TS versions for datasets with multiple anomalies and show a similar performance as DEAN-TS with subsampling for datasets with singular anomalies.

Table 6.8: Anomaly detection performance by dimensionality (AUC-ROC).

Algorithm	Univariate	Multivariate
DEAN-TS-Bag	**0.6854** + 0.0041	0.5784 + 0.0789
DEAN-TS-Bag-RS	0.6245 + 0.0038	0.6466 + 0.0269
DEAN-TS-Bag-SS	0.6286 + 0.0038	0.7342 + 0.0306
EIF	0.5771 + 0.0000	**0.7441** + 0.0310
KNN	0.5917 + 0.0000	0.6114 + 0.0255

Table 6.9: Anomaly detection performance by number of anomalies (AUC-ROC).

Algorithm	1 Anomaly	1 - 5 Anomalies	> 5 Anomalies
DEAN-TS-Bag	**0.6695** + 0.0107	0.6899 + 0.0418	0.6590 + 0.0000
DEAN-TS-Bag-RS	0.5904 + 0.0047	**0.7231** + 0.0212	**0.6710** + 0.0000
DEAN-TS-Bag-SS	0.6270 + 0.0000	0.6959 + 0.0205	0.6445 + 0.0215
EIF	0.6194 + 0.0000	0.6022 + 0.0177	0.5101 + 0.0000
KNN	0.6218 + 0.0000	0.5479 + 0.0161	0.5339 + 0.0000

6.3 Runtime Efficiency

By integrating either of the subsampling approaches, a significant improvement in runtime efficiency was observable. Table 6.10 summarizes the exact runtime measurements, separated by dimensionality and averaged over all datasets. The box plots in Figure 6.2 provide an accompanying visualization. Note that KNN and EIF do not require training in the sense of data-based learning, since KNN is instance-based and EIF is a fully randomized method. Accordingly, only the total runtime is given in both representations.

Table 6.10: Average runtime for the GutenTAG datasets.

Algorithm	Univariate	Multivariate
DEAN-TS-Bag	362.60s	373.63s
DEAN-TS-Bag-RS	152.86s	162.04s
DEAN-TS-Bag-SS	150.02s	159.54s
EIF	5.40s	5.43s
KNN	9.47s	8.62s

Specifically, an approximate halving of the runtime was measured when subsampling was applied for both univariate and multivariate time series. Nevertheless, the application

of DEAN-TS-Bag-SS, as well as DEAN-TS-BAG-RS, still showed a runtime higher than EIF or KNN by a factor of at least 15. At the same time, the remarks from Chapter 5.4 regarding potential drastic improvements for the overall runtime of DEAN-TS by exploiting parallelization still apply. It is also worth noting that there was no significant increase in runtime with increasing dimensionality for any of the algorithms considered.

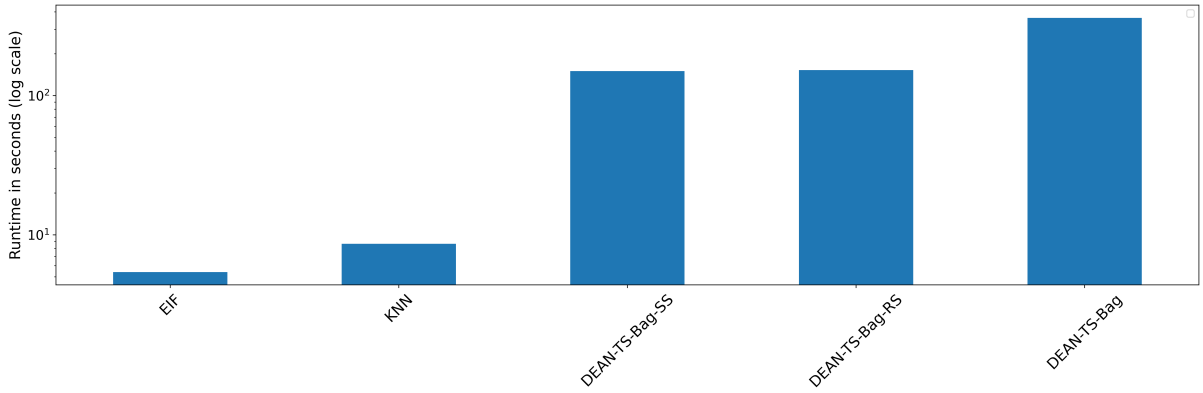


Figure 6.2: Bar chart of the average runtime for the GutenTAG datasets in seconds with log scale.

6.4 Concluding Assessment

In general, many of the tendencies already observed in the semi-supervised benchmarking are confirmed with respect to the different difficulties of anomaly detection for the different base oscillations, anomaly types, dimensionalities, and contamination levels. Here, in the unsupervised setting, the absolute performance scores tend to drop quite a bit though. For a more detailed visual comparison between the results of the two learning paradigms for different aggregation forms of the GutenTAG datasets, see Appendix C.

A simple ranking of the unsupervised performance in this chapter, considering the entire GutenTAG dataset collection for all metrics, can be found in Table 6.11. It is noteworthy that none of the selected competitors made it into the top three in any of these views. At the same time, they were competitive in the detection of trend anomalies and multivariate time series. Furthermore, both methods detected extreme anomaly types significantly better than the DEAN-TS versions, and they also showed consistently better performance for non-periodic base oscillations.

For DEAN-TS-Bag, the use of both subsampling approaches averaged across all datasets seems to be more detrimental to the AUC-ROC performance and has no significant effect on the precision-recall metrics. For the breakdown by individual base oscillations, as well as for the detection of some anomaly types, the loss of information due to subsampling also seems to at least worsen the AUC-ROC performance and generally helps only for

Table 6.11: Average performance ranking across all GutenTAG datasets by each metric.

Metric	Best	2nd Best	3rd Best
AUC-ROC	DEAN-TS-Bag	DEAN-TS-Bag-SS	DEAN-TS-Bag-RS
AUC-PR	DEAN-TS-Bag-RS	DEAN-TS-Bag	DEAN-TS-Bag-SS
AUC-P_TP_T	DEAN-TS-Bag	DEAN-TS-Bag-RS	DEAN-TS-Bag-SS

platform anomalies. Randomized subsampling seems to help, as hoped, with multiple anomalies in the time series, which was not observed with structured subsampling. On the other hand, when considering only multivariate time series, structured subsampling clearly outperformed both other versions of DEAN-TS. Furthermore, the required average runtime by including either subsampling approach was approximately halved.

Due to the very different advantages and disadvantages for different characteristics of the time series and anomalies, it is difficult to make a general recommendation for or against the inclusion of one of the two subsampling variants. The decision should therefore be made depending on the expected characteristics of the use case, but possibly considering larger subsampling sizes in any case to mitigate the seemingly problematic loss of information.

7 Further Experimental Evaluation

After analyzing the overall ensemble performance and runtime under various conditions in the previous two chapters, we now empirically analyze further relevant properties and concepts of DEAN-TS. Specifically, we consider the performance of individual submodels, their variety, and the convergence behavior of the ensemble when they are combined. We then evaluate the usefulness of the thresh method as a combination approach for DEAN-TS by comparing it with alternative methods. Finally, we consider the impact of integrating time series decomposition into Dean-TS in terms of potential performance improvements, changes in ensemble structure, and interpretability.

7.1 Ensemble Structure Analysis

To better understand the final ensemble performance of DEAN-TS and to identify potential improvements, we analyze the results of a semi-supervised application of DEAN-TS at the submodel level. For this purpose, further experiments were performed independently from the benchmark evaluation of Chapter 5 with a parameterization corresponding to the DEAN-TS-Bag variant used there.

For a compact, differentiated evaluation under different, comprehensible preconditions, we restrict ourselves to a subset of the GutenTAG datasets. More precisely, we consider all time series for the two rather complex basic oscillations RW and Poly, in which exactly one single anomaly type was injected, the specific type being indicated by the respective naming of the data sets.

7.1.1 Submodel Performance

Looking at the AUC-ROC performance of individual submodels on their own, divided by base oscillations, as shown in Figure 7.1 and 7.2, it is noticeable that except for the anomaly types amplitude and platform for the RW base oscillation and extremum anomaly for the Poly base oscillation, at least some individual submodels achieve a very good score of > 0.9, but often produce rather mediocre to poor results in median and mean. Nevertheless, it can be noted that even for more complex anomaly types, seemingly expressive relationships can be learned by singular submodels.

While the amplitude anomaly type does not occur in the poly base oscillation due to its intrinsic character, the platform anomaly, a temporally constant value, seems to be much easier to detect than in the case of a RW base oscillation. Remarkably, the structure of the box plots of trend and variance anomalies is exactly reversed between the two base oscillations, while trend anomalies are very well detected by all submodels with a RW base oscillation, with few exceptions.

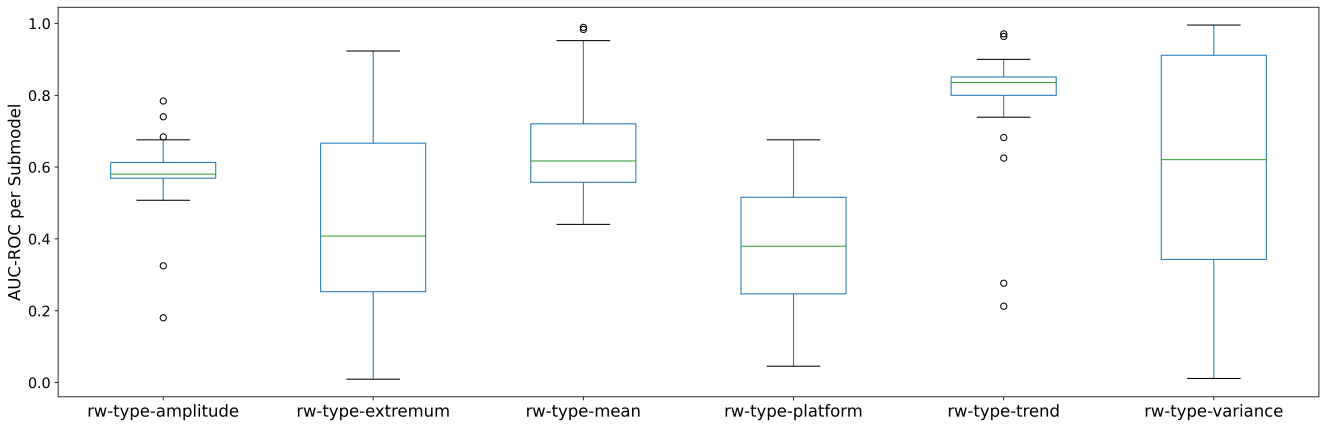


Figure 7.1: Box plots of the AUC-ROC scores of the individual submodels for the different anomaly types of the RW base oscillation.

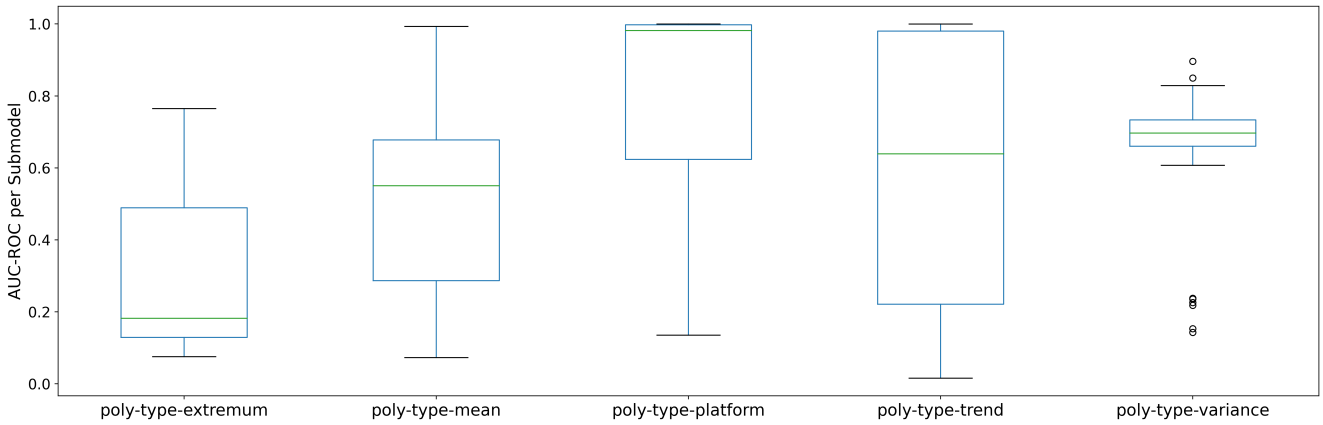


Figure 7.2: Box plots of the AUC-ROC scores of the individual submodels for the different anomaly types of the Poly base oscillation.

7.1.2 Inter-Model Variance and Ensemble Convergence

Since, according to the DEAN properties, the individual submodels do not have to show competitive performance as long as they learn useful, diverse relations and are sensibly combined, we want to take a closer look at the presence of such variability as well as the convergence behavior of the submodel combination when using the thresh method.

For this purpose, the box plots in Figure 7.3 show the standard deviations per data point between the submodel predictions for the considered data sets with RW base oscillation. As a reminder, these predictions correspond to the respective output deviations, normalized to values between 0 and 1, from the q value of the associated MLP. This q value is the average output of the MLP when attempting to map the inputs processed during training to 1. Thus, depending on the type of anomaly, the median of the standard deviations seen in Figure 7.3 generally lies between 5 and 20 percent of the potential deviation range.

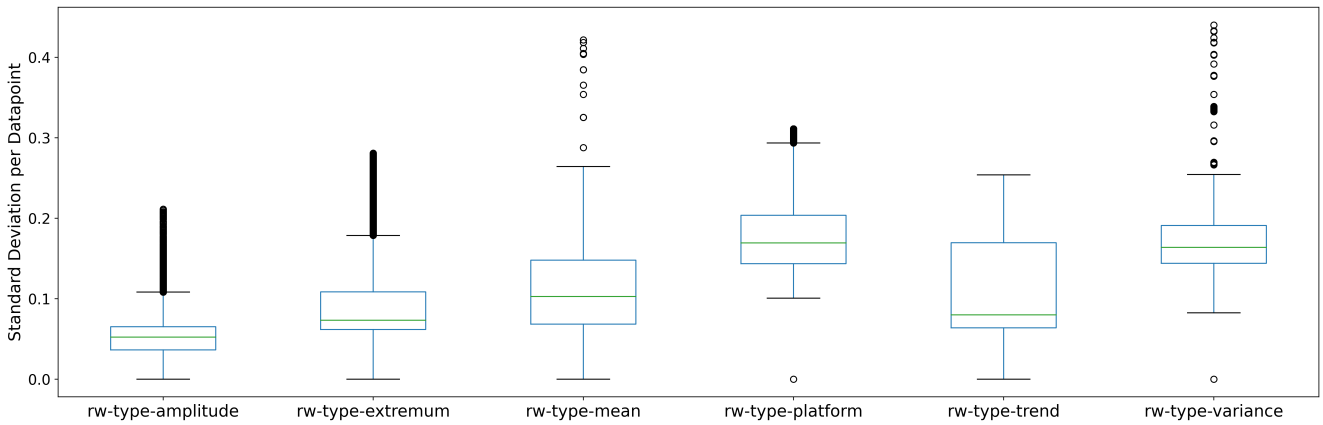


Figure 7.3: Standard deviations for predictions of individual data points between all ensemble submodels for the different anomaly types of the RW base oscillation.

Figure 7.4 shows the convergence behavior of the associated ensembles, contrasting the AUC-ROC ensemble performances as more and more submodels are integrated. Of interest, for example, is the very good AUC-ROC performance of about 0.8565 for the variance anomaly type, although the median of the submodel performances is only about 0.6. For the platform anomaly, the ensemble performance drops very quickly below the median submodel performance and stays at that level. The fluctuations of the ensemble performance decreases sharply starting from an ensemble size of 15. Overall, the performance only changes noticeably for the platform and extremum anomalies, but even here it does not show any more large fluctuations.

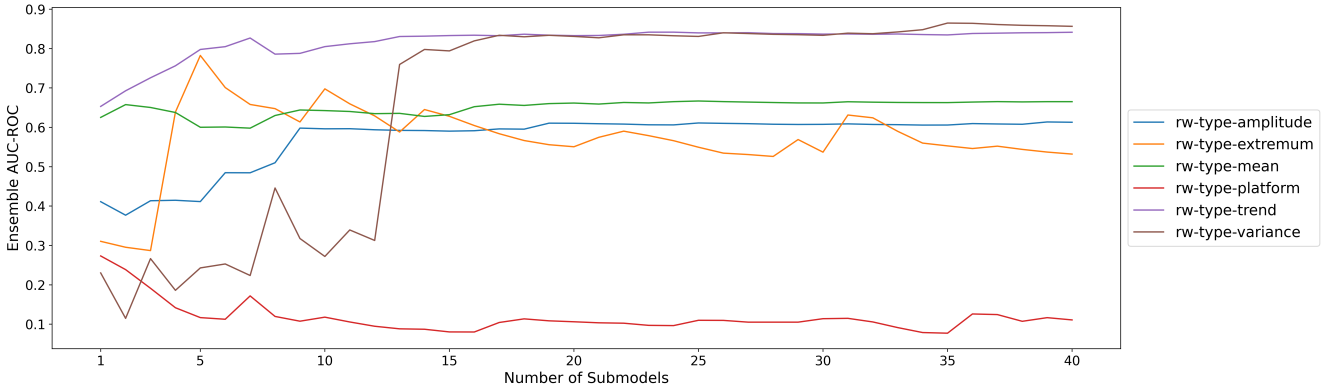


Figure 7.4: Line chart of ensemble AUC-ROC scores with increasing number of included submodels for all anomaly types with RW base oscillation.

The box plots of the standard deviations for the Poly base oscillations in Figure 7.5 show a very similar picture as for the RW counterparts. With regard to the convergence behavior of the corresponding ensembles, shown in Figure 7.6, not much changes from an integration of about 15 submodels onwards, except again for the extremum anomaly.

Moreover, for the mean, extremum, and trend anomalies, there is again a clear increase in performance compared to the median submodel performance, which was originally even higher than the associated mean. Thus, the submodels seem to have a suitable degree of variability, so that their combination can help in learning diverse, robust, and sufficiently complex structures.

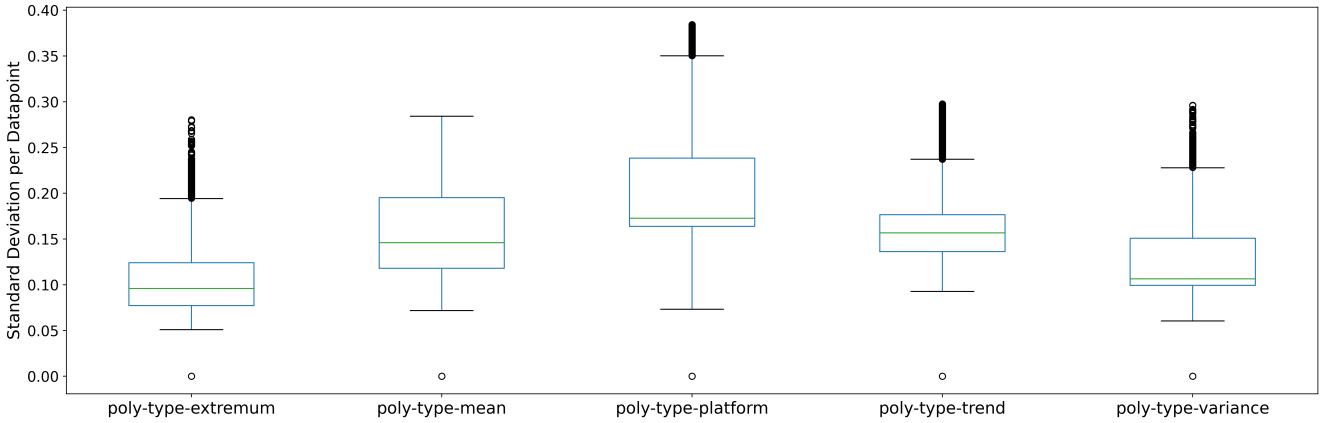


Figure 7.5: Standard deviations for predictions of individual data points between all ensemble submodels for the different anomaly types of the Poly base oscillation.

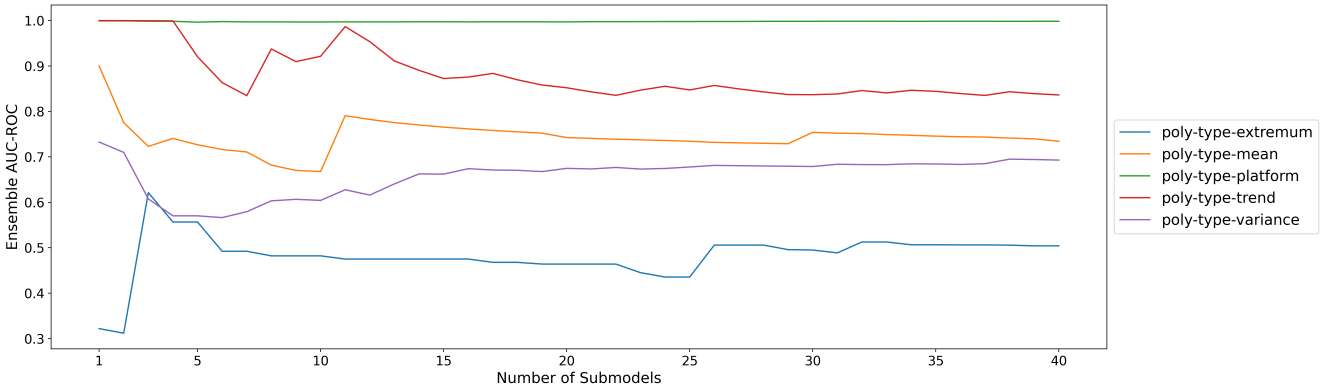


Figure 7.6: Line chart of ensemble AUC-ROC scores with increasing number of included submodels for all anomaly types with Poly base oscillation.

7.2 Influence of the Combination Method

Besides the necessary ability of the submodels to learn complex and diverse relations, a reasonable combination of these submodels is also essential for a good ensemble performance. Chapter 4.2.3 advocates the use of the thresh method, which was applied with a threshold of 0 in both the semi-supervised and unsupervised benchmark evaluations.

In the following, an exemplary comparison with alternative combination methods is made. For this purpose, DEAN-TS-Bag was again applied semi-supervised to all

GutenTAG datasets with a parameterization analogous to the benchmark evaluations. The submodels trained here were additionally combined using the original DEAN combination method of chapter 3.1.2, the standard maximization and average methods, and the thresh method with a threshold of -1. The average AUC-ROC score results, grouped by base oscillation, can be found in table 7.1. A slight deviation of the default method compared to Table 5.5 and Table 5.6 can be observed due to the repetition of the experiments with a different random state.

Table 7.1: Average AUC-ROC score by combination method when applying DEAN-TS-Bag to the GutenTAG datasets (semi-supervised), grouped by base oscillation.

Base Oscillation	Thresh (0)	Thresh(-1)	DEAN	Mean	Max
RW	0.5585	0.5490	0.5583	0.5690	**0.5876**
ECG	0.7827	0.7624	0.7906	0.7810	**0.8057**
CBF	0.5506	0.5552	0.5717	**0.5787**	0.5152
Poly	0.6189	0.6120	0.5869	0.5850	**0.6340**
Sinus	0.9709	0.9693	0.9707	**0.9716**	0.9587

When grouped in this way, it is noticeable that for each base oscillation, either the conventional averaging or the maximization combination produced the best results, albeit by a narrow margin. However, when the methods are compared pairwise, there is no clear ranking. As can be seen in the box plots in Figure 7.7 for the RW base oscillation, this similarity of performance extends across the distribution of good to poor performance, with the variance for maximization being somewhat lower than for the other methods.

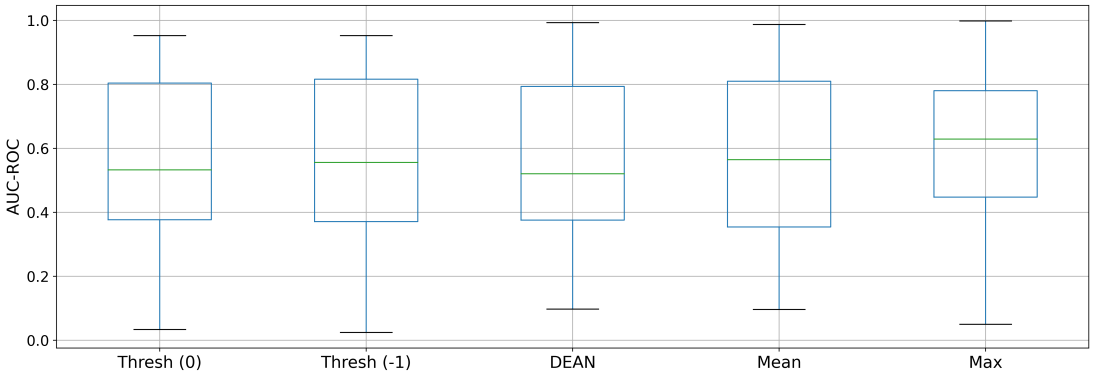


Figure 7.7: Box plots of AUC-ROC scores across all GutenTAG datasets with RW base oscillation by combination function.

Although maximization seems to be slightly ahead for the RW base oscillations at first glance, a closer analysis shows that this combination method has a higher risk of poor performance [2]. Looking at the list of frequencies in Figure 7.8, which shows how often a

combination method has achieved the best and the worst result, maximization is on top in both categories. The other methods all appear to be more balanced in this regard, with the average combination method looking particularly attractive in this regard.

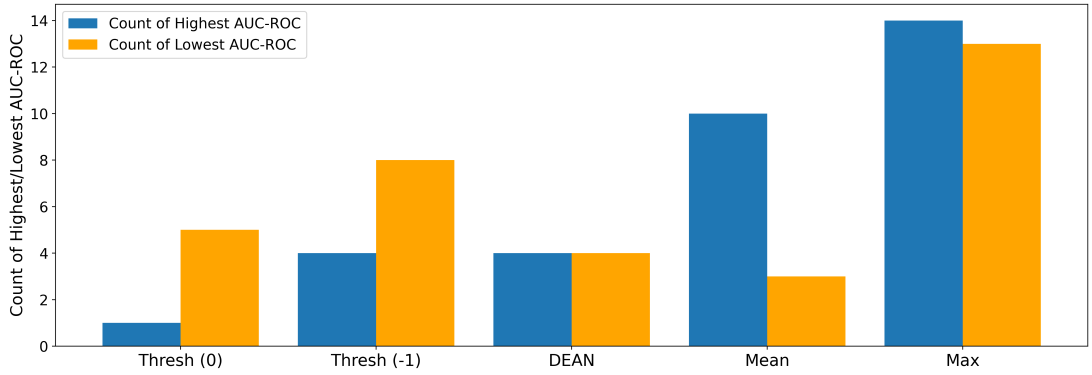


Figure 7.8: Bar chart of how often which combination function achieved the lowest and highest AUC-ROC score on the GutenTAG datasets with RW base oscillation.

Overall, the choice of combination method does not seem to have a significant effect, at least on the average AUC-ROC performance. However, the clearer delineation of anomalous and normal data points of the thresh method with a threshold of zero, as mentioned in Chapter 4.2.3, could be observed repeatedly and could thus facilitate the interpretation for a human observer.

7.3 Incorporating Time Series Decomposition

In the following, we consider the results of an experimental integration of time series decomposition (TSD) using STL as a preprocessing step in DEAN-TS. Due to the excellent identifiability of the different anomaly types for the analysis, we again focus on the GutenTAG data sets, more precisely on those with ECG base oscillations. Their periodic character matches the requirements for STL (see chapter 4.3) and left more room for improvement in the benchmark evaluations than the sinusoidal data sets. Again, the experiments were semi-supervised and performed throughout with the DEAN-TS-Bag parameterization, although for the analyses in subsections 7.3.3 and 7.3.4 the ensemble size was increased from 40 to 70 for more robust results.

7.3.1 Potential for Improvement

A visualization of the potential performance gain in anomaly detection can be found in Figure 7.10. Here we see the results of applying DEAN-TS-BAG with 20 submodels with and without TSD as a preprocessing step on the so-called ecg-noise-10% dataset. This is

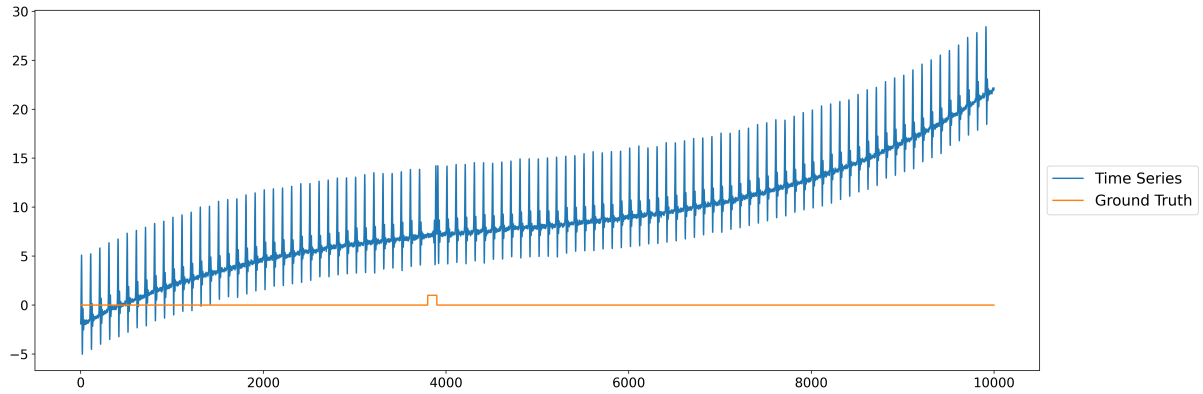


Figure 7.9: Visualization of the ecg-noise-10% time series with ground truth.

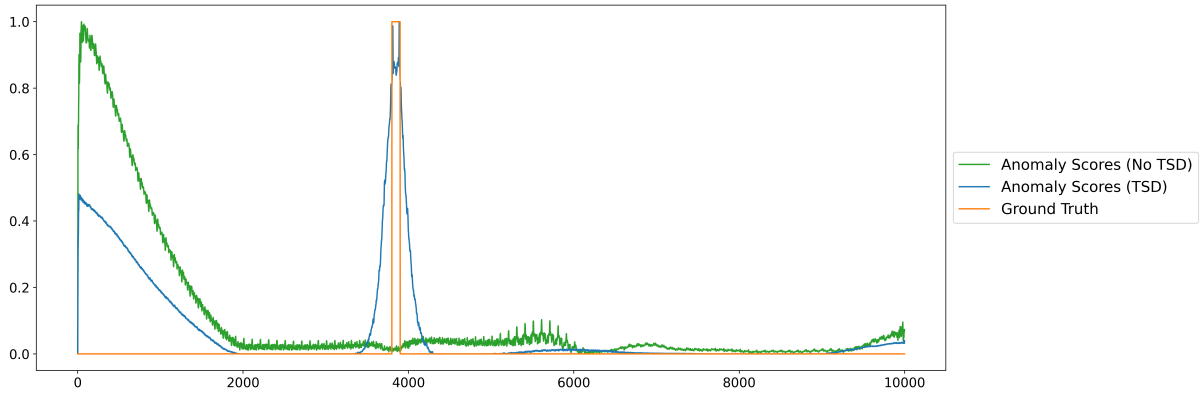


Figure 7.10: Predictions for the ecg-noise-10% time series with and without TSD.

a univariate time series with a general underlying polynomial trend and a single pattern type anomaly. Without the integration of TSD, the anomaly is not detected at all, as reflected by the associated AUC-ROC score of 0.2479. With TSD, however, the anomaly is accurately detected, resulting in a perfect AUC-ROC score of 1.0.

As seen in Table 7.2, this trend can be observed across various datasets. Here we list the differences in AUC-ROC performance and runtime when TSD is integrated for those ECG datasets where DEAN-TS-Bag initially achieved an AUC-ROC score of <0.4 in the benchmark evaluation of Chapter 5.2. Since the experiments were repeated with a different random state, the results for the variant without TSD differ slightly from the initial benchmark results. It can be noted that with the exception of the ecg-type trend dataset, which has only a single trend anomaly, the AUC-ROC performance was significantly increased throughout. Similar results regarding the generally remarkable increase of the AUC-ROC performance could also be seen in random experiments for the periodic NASA time series considered in Chapter 5.3.1.

Notably, the increase in runtime is not due to preprocessing, i.e. computing the decomposition, which took about 1 second on average, but to a subsequent increase in

Table 7.2: Comparison of AUC-ROC performance and runtime in seconds with and without preprocessing by means of applying time series decomposition to the initial time series.

Algorithm	No TSD		TSD	
	AUC-ROC	Runtime	AUC-ROC	Runtime
ecg-combined-diff-1	0.3612	145.87s	**0.8941**	149.85s
ecg-length-10	0.1700	126.43s	**0.9999**	208.76s
ecg-length-100	0.3897	127.53s	**0.9997**	214.82s
ecg-noise-10%	0.6343	146.89s	**0.9997**	158.73s
ecg-trend-sinus	0.1543	158.08s	**1.0**	245.65s
ecg-type-trend	**0.3893**	137.44s	0.3428	232.91s

training time, which was quite substantial for ecg-length-10, ecg-length-100, ecg-trend-sinus and ecg-type-trend. For the others, however, there was no increase in training time at all. The slight increase in total runtime for them was only due to the combination of preprocessing and a slight increase in prediction time. While the TSD application naturally increases the dimensionality, the fact that this does not seem to increase the runtime by a relevant amount is consistent with the results from Chapter 5.4.

7.3.2 Changes in Ensemble Structure

The usually strong increase in AUC-ROC scores is also evident at the level of individual submodels. A comparison of the corresponding box plots in Figure 7.11 and 7.12 shows a strong increase in median performance almost everywhere. The only exception is the ecg-type-trend dataset, where the overall performance has decreased, as already mentioned. Interestingly, the number of submodels with performance well above the median has increased significantly. In general, the time series decomposition seems to facilitate the variability of the performance, which can be seen in the larger distances between the quantile values.

An analysis of the AUC-ROC score development of DEAN-TS with increasing number of submodels in the Figures 7.13 and 7.14 also shows that the usually much better ensemble performance with the integration of TSD has mostly stabilized with only 5 submodels. It is also notable that the more acceptable performance for the ecg-type-trend dataset of the DEAN-TS version without TSD developed abruptly after training about 40 submodels.

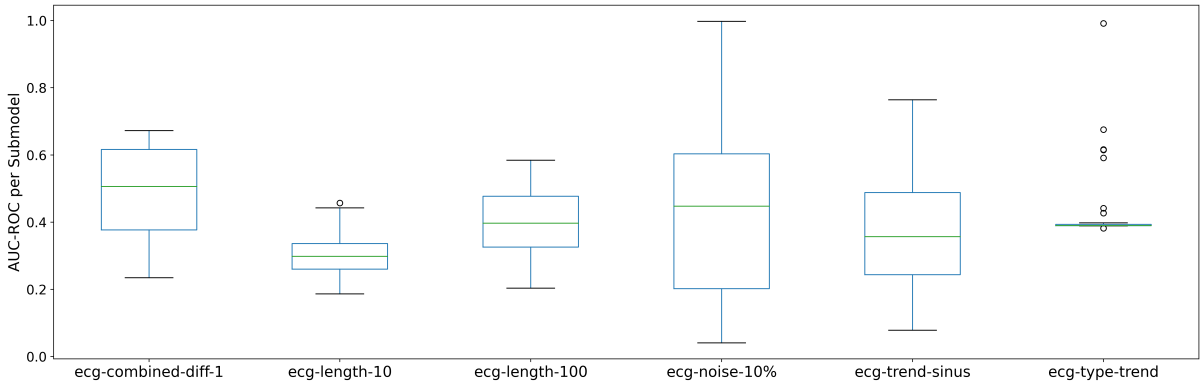


Figure 7.11: Box plots of the AUC-ROC scores of the individual submodels (No TSD).

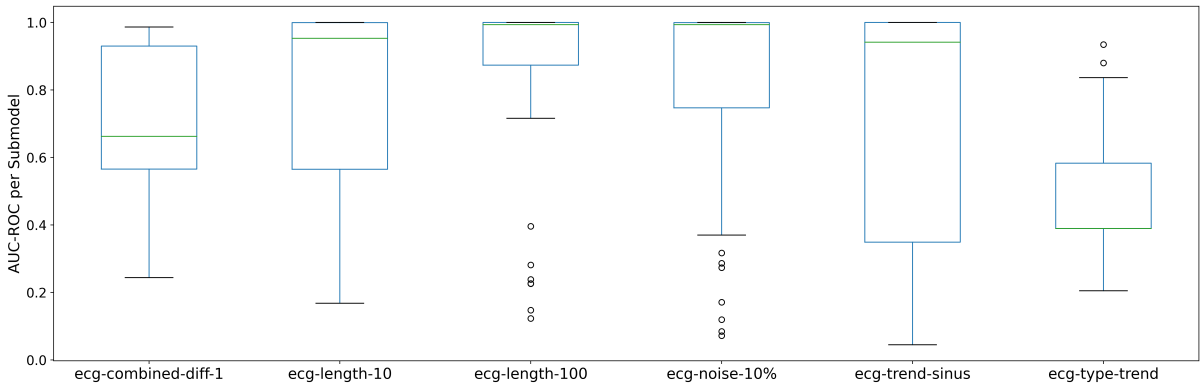


Figure 7.12: Box plots of the AUC-ROC scores of the individual submodels (TSD).

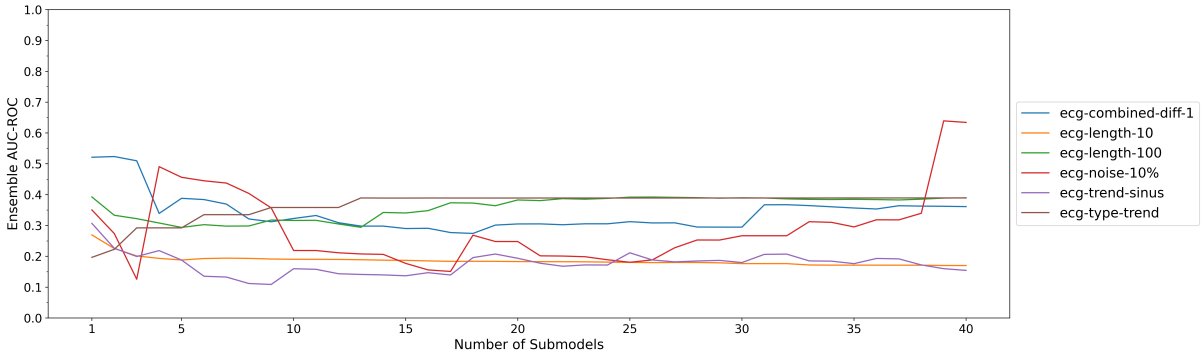


Figure 7.13: Line chart of ensemble AUC-ROC scores with increasing number of included submodels (No TSD).

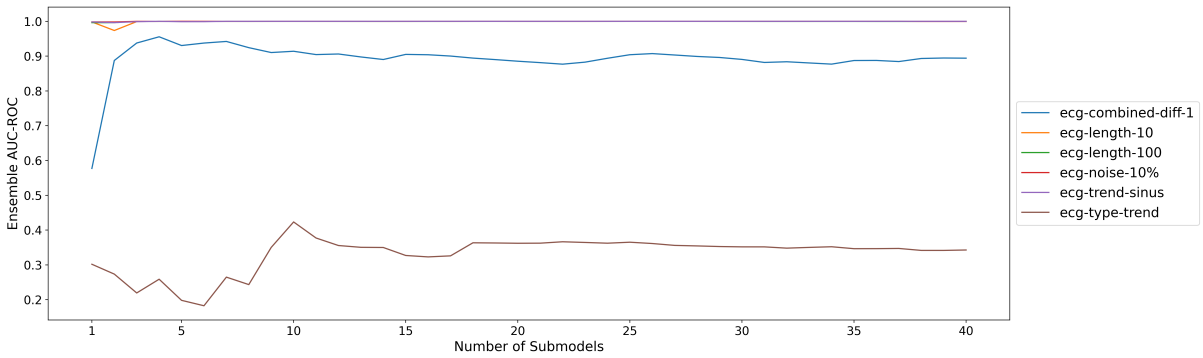


Figure 7.14: Line chart of ensemble AUC-ROC scores with increasing number of included submodels (TSD).

7.3.3 Impact of Feature Selection

Integrating different time series components by feature bagging increases the variability and allows to focus on structural aspects, but depending on the characteristics of the time series and the anomaly types, some models perform worse in terms of AUC-ROC score performance. It should be noted that the learned relations can still be useful for anomalies that have not been encountered. This can be visualized by analyzing the submodel performance depending on the time series components selected by feature bagging for different anomaly types. According to the interpretability approach described in chapter 4.4, we consider the performance depending on all possible feature tuples, since the interaction of the individual components is a relevant factor.

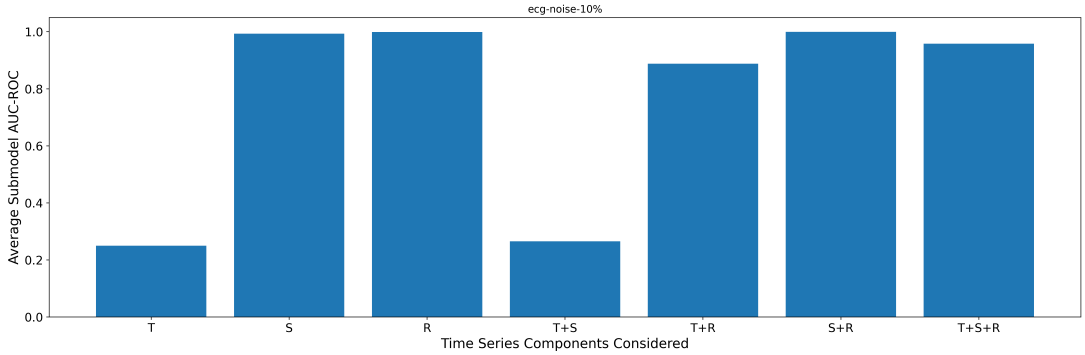


Figure 7.15: Average submodel performance by TSD components considered for ecg-noise-10%

The bar plots in Figures 7.15 and 7.16 show such a breakdown of submodel performance by included components, with the trend, seasonality, and residual components indicated by the first letter of their names. For the pattern anomaly of the ecg-noise-10% dataset, it is clearly visible, that the inclusion of the trend component is rather detrimental, while all submodels that include the residual component perform very well on average. On the other

hand, for the detection of the mean anomaly in the ecg-type-mean data set, the residual component by itself does not seem to be helpful. Furthermore, in the ecg-type-trend dataset, which as shown in Table 7.2 is the only one that still performed poorly when TSD was applied, here only the submodels that solely considered the residual component achieved an average AUC-ROC of > 0.8.

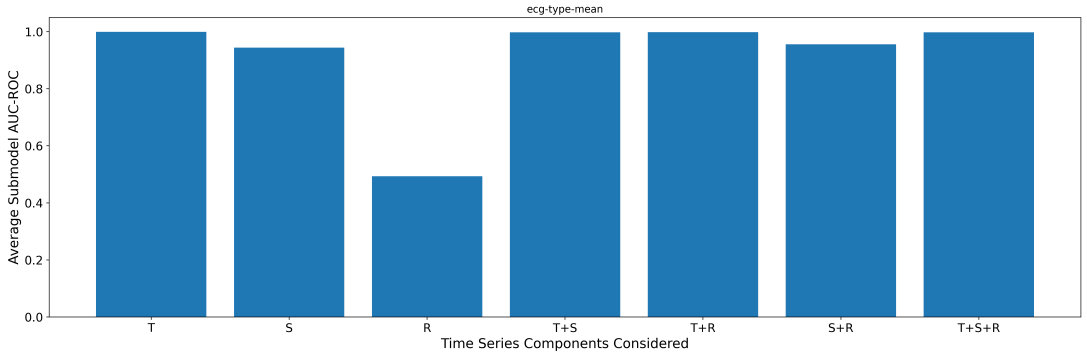


Figure 7.16: Average submodel performance by TSD components considered for ecg-type-mean.

7.3.4 Feature Importance Interpretability

Finally, we want to experimentally evaluate the DEAN-TS interpretability approach based on the importance of all possible TSD component combinations. For this purpose, we consider the ecg-diff-count-9 dataset shown in Figure 7.17. When generating this high frequency ECG time series, each of the presented GutenTAG anomaly types was injected once in a non-overlapping manner. While the detection performance as shown in Figure 7.18 is already comparatively good without TSD, it improves even further with the additional preprocessing, so that for example the extremum anomaly is now also detected.

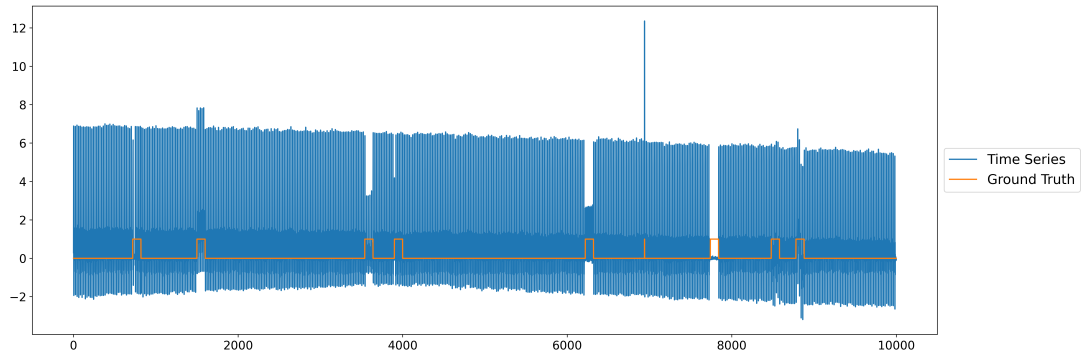


Figure 7.17: Time series values next to the corresponding ground truth regarding anomalies for the ecg-diff-count-9 dataset.

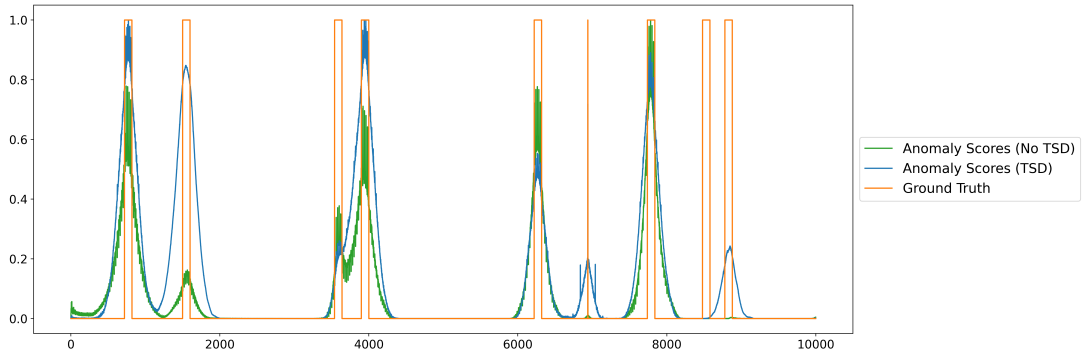


Figure 7.18: Difference in Ensemble prediction with and without TSD preprocessing.

As can be seen in Figure 7.19, due to the different types of anomalies, a large number of the relations learned by different TSD components were violated, which is probably why each combination of the components alone already gave decent results.

To assess the importance of the TSD components without ground truth for some of the (correctly) identified anomaly subsequences, we now consider the importance scores formalized in chapter 4.4. For this we restrict ourselves to the extremum anomaly (time step 6940) and the trend anomaly (time steps 8780-8880), both of which could only realiably be identified by the DEAN-TS version using TSD, as can be seen in Figure 7.18.

For the trend anomaly, we calculate the averaged importance score over all affected data points. As a reminder, this score is obtained for a TSD component combination with respect to a single data point as the difference between the submodel predictions using that combination as a feature set and the average submodel prediction. The results are visualized in Figure 7.20, where each combination of the components trend, seasonality, and residual possible by feature-bagging with one to three features is again identified by the initial letters of all included components.

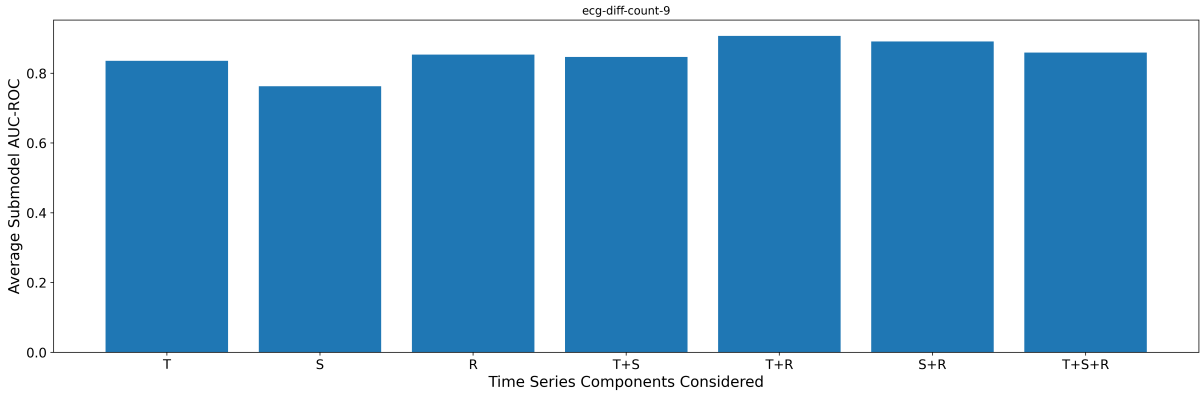
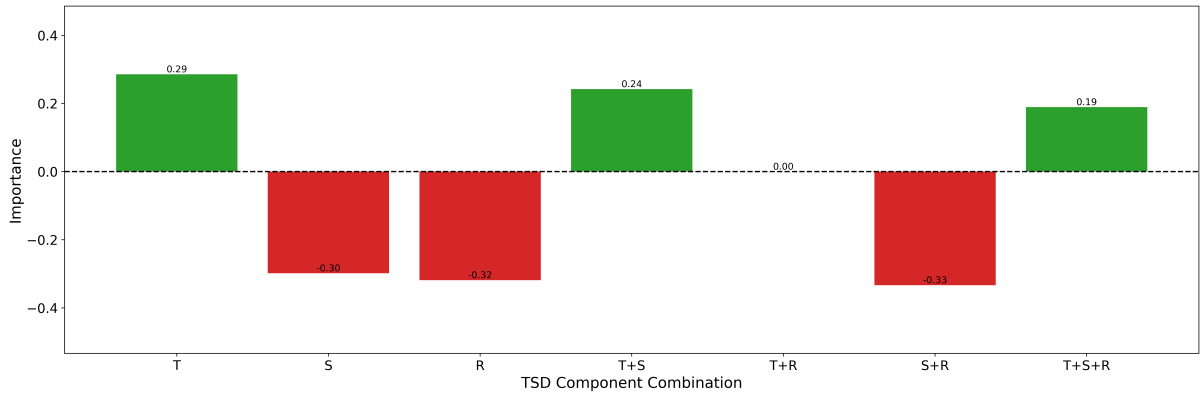
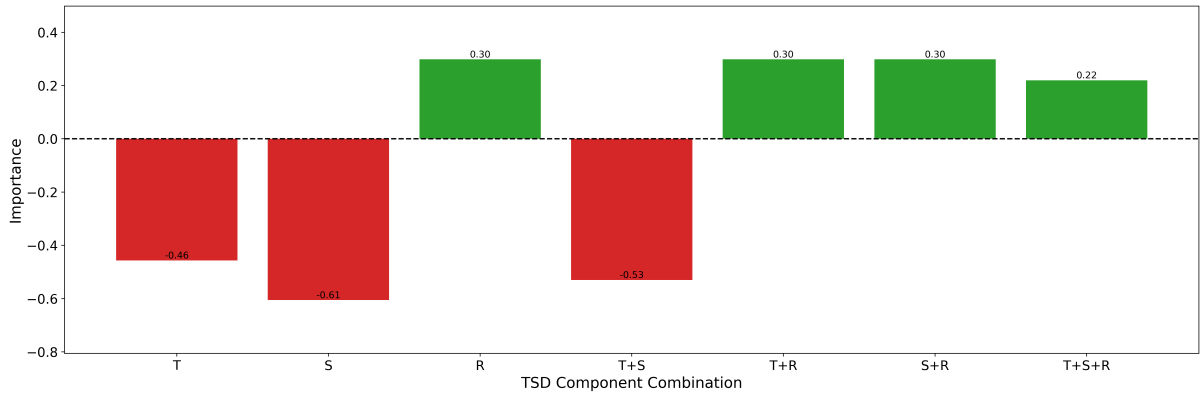


Figure 7.19: Average submodel performance by TSD components considered for ecg-diff-count-9.

Note that the importance for the different anomaly types is varying significantly. For example, when learning relations to identify a trend anomaly, it seems useful, or rather necessary, to select the trend component as a feature, either by itself or in combination with the seasonality component. The seasonality component alone does not seem to be sufficient for this purpose, and considering the residual component even seems to be rather detrimental. However, for the detection of an extreme anomaly, the residual component is of utmost importance. By adding another TSD component, the submodels apparently detect this type of anomaly as well as before, but when all three components are included, it seems to become less clear.



(a) TSD component combination importance for trend anomaly.



(b) TSD component combination importance for extremum anomaly.

Figure 7.20: Importance scores for the individual TSD component combinations for different anomalies in the ecg-diff-count-9 dataset. Each potential combination is indicated by the initials of all selected components in the given combination.

8 Conclusion and Outlook

In this thesis, the concepts of the DEAN method have been transferred to the time series domain where they have been analyzed in detail. For this purpose, an adapted methodology, called DEAN-TS, has been developed, which takes into account the peculiarities of temporal data by means of various adaptations, while still adhering to the DEAN properties. The submodel architecture of DEAN is retained, so the emphasis here is on preprocessing the input data to reveal temporal dependencies using a window-based approach. Furthermore, the original combination method is replaced by the tresh method, which favors a clearer delineation between anomalous and normal data points, as well as a balance between being more bias- or variance-aware. Furthermore, the integration of two different subsampling approaches has been proposed, as well as time series decomposition using STL. For the latter, an adapted version of the DEAN interpretability method was developed.

The DEAN-TS method has consequently been evaluated for its anomaly detection performance and runtime efficiency in different parameterizations on a variety of synthetic and real world time series datasets, both semi-supervised and unsupervised, and compared with related methods in this respect. For the synthetic datasets, it was shown that DEAN-TS can achieve very good anomaly detection performance even without integrating the time series decomposition, especially for time series with periodic underlying oscillations and selected anomaly types, but that it often lags behind related methods. On the other hand, it repeatedly showed competitive performance on the real world data sets.

A rather restrictive parameterization with respect to the theoretical submodel capabilities, such as more layers or the addition of a bias term, has, except in exceptional cases, hurt rather than helped, while DEAN-TS has, on average, achieved comparable results with respect to different parameter options. The integration of the proposed subsampling approaches in the unsupervised setting improved the anomaly detection performance in some cases, but often harmed it, although a significant improvement in runtime efficiency was generally observed. Overall, the runtime efficiency was average compared to related semisupervised methods in the configurations considered, but has much room for improvement due to the large parallelization potential.

By further analysis of the ensemble structure, it was observed that individual submodels can already learn expressive relations by themselves and that the variability of the submodels is usually comparatively high. Furthermore, the choice of the combination method does not seem to have a major impact on the average AUC-ROC score performance. On the other hand, the integration of the time series decomposition resulted in extreme performance gains for almost all periodic time series that initially showed rather

poor results in the benchmark evaluation. Furthermore, positive effects on ensemble convergence speed and stability were observed. In an experimental application of the interpretability method based on time series decomposition, comprehensible results were observed for the selected anomaly types. In conclusion, the concepts of DEAN combined with appropriate preprocessing show potential for application in the time series domain, especially when coupled with time series decomposition.

To even more effectively fulfill the objectives formulated in Chapter 4.1, there are several directions that seem to merit future work. Since promising results have been obtained especially in combination with time series decomposition, the robust integration of this concept also on time series with chaotic structures seems particularly promising. Fortunately, there is already research on more robust versions of e.g. the used STL method [47].

Regarding runtime efficiency, the current design of the method is likely to achieve extreme gains by parallelizing the implementation, so the rather average results of the runtime evaluation should not be overestimated [26]. Conceptually, the current prediction variance of the submodels could be monitored by means of test data, so that no further submodels are trained at an early stage if the prediction quality appears stable, similar to early stopping, but on the ensemble level.

The already semantically expressive importance scores computed by the developed interpretability method could be used to provide a high-level interpretation of the predictions by means of observed relationships with different anomaly types. Depending on the use case preferences regarding the importance of different anomaly types, an adjustment according to the perceived importance could even be performed as a post-processing step. In order to discretize the output scores into more manageable binary decisions for human interpreters, an automatic threshold could be computed, analogous to what has already been proposed for Telemanom [21] or OmniAnomaly [45].

References

[1] Aggarwal, C. C. (2017). *Outlier Analysis*. Springer.

[2] Aggarwal, C. C. and Sathe, S. (2015). Theoretical foundations and algorithms for outlier ensembles. *ACM SIGKDD Explorations Newsletter*, 17(1):24–47.

[3] Aggarwal, C. C. and Sathe, S. (2017). *Outlier ensembles: An introduction*. Springer.

[4] Böing, B., Klüttermann, S., and Müller, E. (2022). Post-robustifying deep anomaly detection ensembles by model selection. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 861–866. IEEE.

[5] Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

[6] Braei, M. and Wagner, S. (2020). Anomaly detection in univariate time-series: A survey on the state-of-the-art. *arXiv preprint arXiv:2004.00433*.

[7] Chauhan, S. and Vig, L. (2015). Anomaly detection in ecg time signals via deep long short-term memory networks. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pages 1–7. IEEE.

[8] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[9] Cleveland, R. B., Cleveland, W. S., McRae, J. E., and Terpenning, I. (1990). Stl: A seasonal-trend decomposition. *J. Off. Stat*, 6(1):3–73.

[10] Darban, Z. Z., Webb, G. I., Pan, S., Aggarwal, C. C., and Salehi, M. (2022). Deep learning for time series anomaly detection: A survey. *arXiv preprint arXiv:2211.05244*.

[11] Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240.

[12] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

[13] Faraway, J. and Chatfield, C. (1998). Time series forecasting with neural networks: a comparative study using the air line data. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 47(2):231–250.

[14] Gao, J., Song, X., Wen, Q., Wang, P., Sun, L., and Xu, H. (2020). Robusttad: Robust time series anomaly detection via decomposition and convolutional neural networks. *arXiv preprint arXiv:2002.09545*.

[15] Golmohammadi, K. and Zaiane, O. R. (2015). Time series contextual anomaly detection for detecting market manipulation in stock market. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10.

[16] Hanley, J. A. and McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36.

[17] Hariri, S., Kind, M. C., and Brunner, R. J. (2019). Extended isolation forest. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1479–1489.

[18] Hawkins, D. M. (1980). *Identification of outliers*, volume 11. Springer.

[19] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

[20] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[21] Hundman, K., Constantinou, V., Laporte, C., Colwell, I., and Soderstrom, T. (2018). Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 387–395.

[22] Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.

[23] Kadous, M. W. (1999). Learning comprehensible descriptions of multivariate time series. In *ICML*, volume 454, page 463.

[24] Kingma, D. P. and Welling, M. (2014). Stochastic gradient vb and the variational auto-encoder. In *Second international conference on learning representations, ICLR*, volume 19, page 121.

[25] Klüttermann, S. and Mülller, E. (2022). Dean: Deep ensemble anomaly detection. `https://github.com/KDD-OpenSource/DEAN/`.

[26] Kriegel, H.-P., Schubert, E., and Zimek, A. (2017). The (black) art of runtime evaluation: Are we comparing algorithms or implementations? *Knowledge and Information Systems*, 52:341–378.

[27] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

[28] Lazarevic, A. and Kumar, V. (2005). Feature bagging for outlier detection. volume 21, pages 157–166.

[29] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[30] Li, Z., Zhao, Y., Han, J., Su, Y., Jiao, R., Wen, X., and Pei, D. (2021). Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 3220–3230.

[31] Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE.

[32] Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, Georgia, USA.

[33] Munir, M., Chattha, M. A., Dengel, A., and Ahmed, S. (2019). A comparative analysis of traditional and deep learning-based anomaly detection methods for streaming data. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 561–566.

[34] Ogbechie, A., Díaz-Rozo, J., Larrañaga, P., and Bielza, C. (2017). Dynamic bayesian network-based anomaly detection for in-process visual inspection of laser surface heat treatment. In *Machine Learning for Cyber Physical Systems: Selected papers from the International Conference ML4CPS 2016*, pages 17–24. Springer.

[35] Pantiskas, L., Verstoep, K., and Bal, H. (2020). Interpretable multivariate time series forecasting with temporal attention convolutional neural networks. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1687–1694. IEEE.

[36] Pereira, J. and Silveira, M. (2019). Learning representations from healthcare time series data for unsupervised anomaly detection. In *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 1–7.

[37] Raghavan, V., Bollmann, P., and Jung, G. S. (1989). A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems (TOIS)*, 7(3):205–229.

[38] Ramaswamy, S., Rastogi, R., and Shim, K. (2000). 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.

[39] Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR.

[40] Rodriguez, A. C. and de los Mozos, M. R. (2010). Improving network security through traffic log anomaly detection using time series analysis. In *Computational Intelligence in Security for Information Systems 2010: Proceedings of the 3rd International Conference on Computational Intelligence in Security for Information Systems (CISIS'10)*, pages 125–133. Springer.

[41] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[42] Sakurada, M. and Yairi, T. (2014). 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.

[43] Schmidl, S., Wenig, P., and Papenbrock, T. (2022). Anomaly detection in time series: a comprehensive evaluation. *Proceedings of the VLDB Endowment*, 15(9):1779–1797.

[44] Song, H., Jiang, Z., Men, A., and Yang, B. (2017). A hybrid semi-supervised anomaly detection model for high-dimensional data. *Computational intelligence and neuroscience*, 2017.

[45] Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., and Pei, D. (2019). 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.

[46] Tatbul, N., Lee, T. J., Zdonik, S., Alam, M., and Gottschlich, J. (2018). Precision and recall for time series. *Advances in neural information processing systems*, 31.

[47] Wen, Q., Gao, J., Song, X., Sun, L., Xu, H., and Zhu, S. (2019). Robuststl: A robust seasonal-trend decomposition algorithm for long time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5409–5416.

[48] Wenig, P., Schmidl, S., and Papenbrock, T. (2022). Timeeval: a benchmarking toolkit for time series anomaly detection algorithms. *Proceedings of the VLDB Endowment*, 15(12):3678–3681.

[49] Zhang, L., Shen, X., Zhang, F., Ren, M., Ge, B., and Li, B. (2019). Anomaly detection for power grid based on time series model. In *2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, pages 188–192.

[50] Zimek, A., Schubert, E., and Kriegel, H.-P. (2012). A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(5):363–387.

Appendix A Implementation Details

The implementation of DEAN-TS was developed in Python 3.10 using some of the most prominent libraries for Data Science applications. In particular, *Numpy*[2], *TensorFlow*[3], *Keras*[4], *SciPy*[5] and *statsmodels*[6] were utilized. During development, emphasis was placed on a modular, object-oriented implementation and a declarative approach was taken for parameterization, using a .yaml configuration file. For the benchmark evaluations, the integration using TimeEval was implemented similarly to the related methods using docker containerization according to the interface requirements defined in the TimeEval documentation[7]. The source code is available at: `https://github.com/Ti-Kat/dean-ts`.

[2] `https://www.numpy.org/`
[3] `https://www.tensorflow.org/`
[4] `https://www.keras.io/`
[5] `https://www.scipy.org/`
[6] `https://www.statsmodels.org/`
[7] `https://www.timeeval.readthedocs.io/`

Appendix B Algorithm Parameterizations

The following tables specify the parameterizations used for the various algorithms applied throughout this work. Table A2.1 first describes the parameterization underlying all DEAN-TS versions as default under the entry DEAN-TS, specified according to the actual implementation referenced in Appendix A. For the sake of conciseness, the subsequent entries list only the parameters modified in the corresponding version. Table A2.2 and A2.3 specify the parameter choices for the related work used in Chapter 5 and 6 for the benchmark evaluations. These match the default values proposed in the respective implementations[8].

Table A2.1: Parameterization for DEAN-TS and version-specific adjustments.

Algorithm	Parameter	Value
DEAN-TS	ensemble_size	40
	tsd	False
	reverse_window	True
	combination_method	'thresh'
	thresh_value	0
	feature_bagging	True
	fb_range	[1,3]
	subsampling	'none'
	bias	False
	depth	3
	activation	'relu'
	lag_indices _count	127
	look_back_range	[128, 1024]
	lr	0.01
	batch	32
	random_state	21
DEAN-TS-Bag	lag_indices _count	63
	look_back_range	[64, 512]
DEAN-TS-Bias	bias	True
DEAN-TS-Depth	depth	4
DEAN-TS-Leaky	activation	'leaky'
DEAN-TS-Bag-RS	subsampling	'random'
	rs_range	[1024, 8192]
DEAN-TS-Bag-SS	subsampling	'structured'
	ss_r	[0.1, 0.25]
	ss_m	[2, 5]

[8] https://github.com/HPI-Information-Systems/TimeEval-algorithms

Table A2.2: Parameterizations for related methods (semi-supervised).

Algorithm	Parameter	Value
Hybrid KNN [44]	linear_layer_shape	(100,10)
	split	0.8
	anomaly_window_size	20
	batch_size	64
	test_batch_size	256
	epochs	1
	early_stopping_delta	0.05
	early_stopping_patience	10
	learning_rate	0.001
	n_neighbors	12
	n_estimators	3
	random_state	42
LaserDBN [34]	timesteps	2
	n_bins	10
	random_state	42
OmniAnomaly [45]	latent_size	3
	rnn_hidden_size	500
	window_size	100
	linear_hidden_size	500
	nf_layers	20
	epochs	10
	split	0.8
	batch_size	50
	l2_reg	0.0001
	learning_rate	0.001
	random_state	42
Telemanom [21]	batch_size	70
	smoothing_window_size	30
	smoothing_perc	0.05
	error_buffer	100
	dropout	0.3
	lstm_batch_size	64
	epochs	35
	split	0.8
	early_stopping_patience	10
	early_stopping_delta	0.0003
	window_size	250
	prediction_window_size	10
	p	0.13
	random_state	42

Table A2.3: Parameterizations for related methods (unsupervised).

Algorithm	**Parameter**	**Value**
EIF [17]	n_trees	200
	max_samples	None
	extension_level	None
	limit	None
	random_state	42
KNN [38]	n_neighbors	5
	leaf_size	30
	method	largest
	radius	1.0
	distance_metric_order	2
	n_jobs	1
	random_state	42

Appendix C Additional Benchmark Plots

The following plots give an additional visualization of the average observed AUC-ROC performance of the different DEAN-TS versions when applied to the GutenTAG datasets from the benchmark evaluation chapters 5 and 6. For this purpose, the datasets are grouped or filtered by base oscillation (Figure A3.1 and A3.2), anomaly type (Figure A3.3 and A3.4), as well as dimensionality and contamination (Figure A3.5 and A3.6), analogous to the analysis these chapters. For each grouping there is a figure for the semi-supervised and unsupervised setting. The DEAN-TS versions applied there are compared with each other and with the best observed performance according to the currently considered criteria.

In most cases, the unsupervised setting shows a relevant drop in the best observed performance. Especially the direct comparison of DEAN-TS-Bag for an application with and without anomalies in the training data set allows an estimation about the influence of this factor depending on the characteristics of the corresponding time series, as well as the number and type of anomalies.

C.1 Anomaly Detection Performance by Base Oscillation

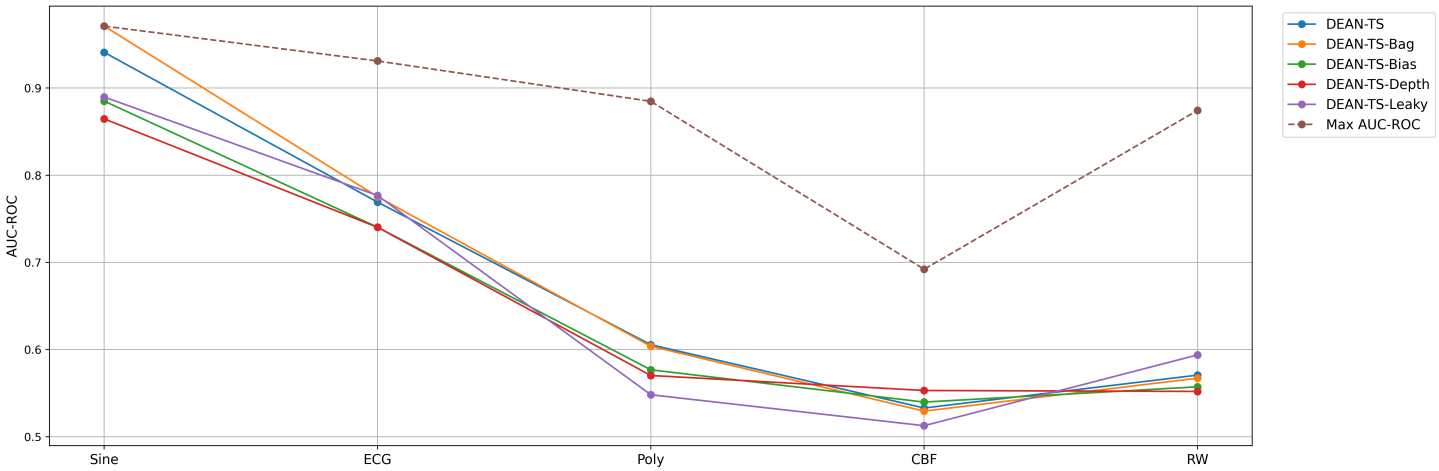


Figure A3.1: Average performance by base oscillation (semi-supervised). Compares DEAN-TS with the best observed performance.

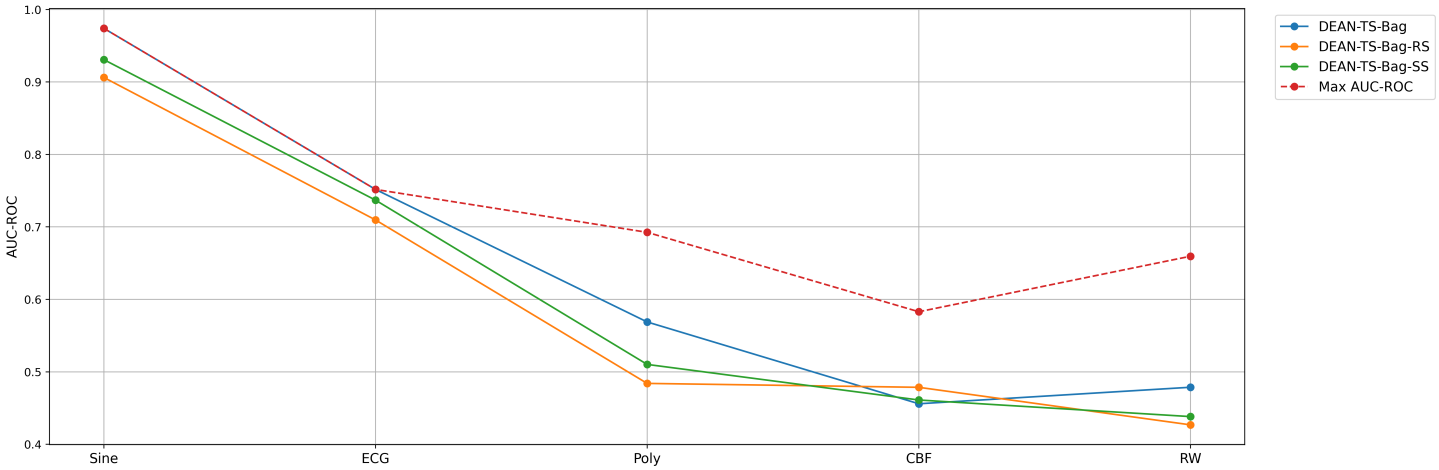


Figure A3.2: Average performance by base oscillation (unsupervised). Compares DEAN-TS with the best observed performance.

C.2 Anomaly Detection Performance by Anomaly Type

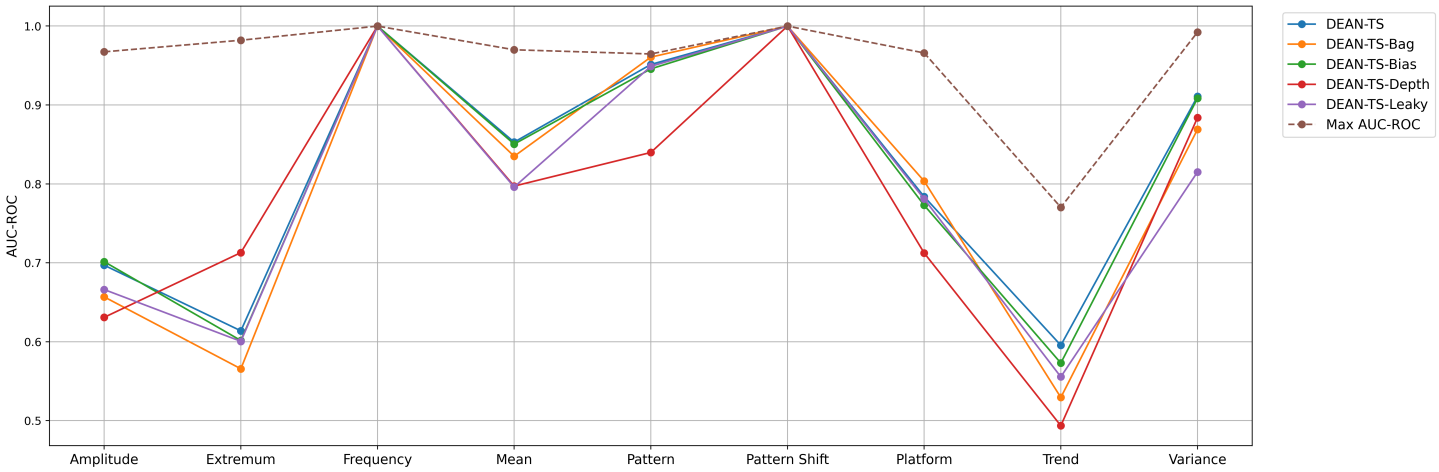


Figure A3.3: Average performance by anomaly type (semi-supervised). Compares DEAN-TS with the best observed performanc .

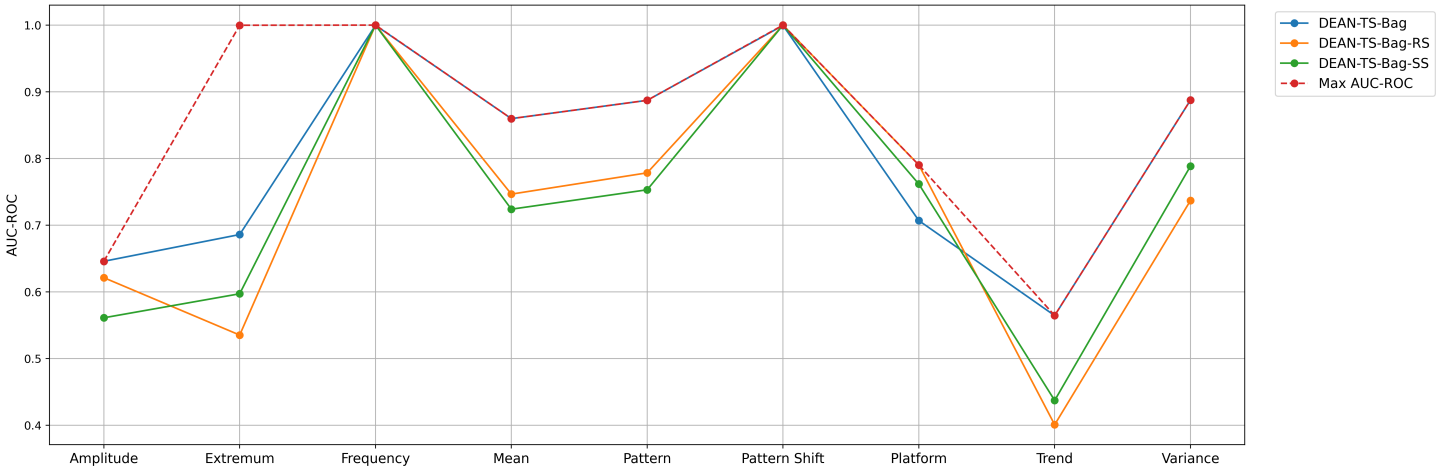


Figure A3.4: Average performance by anomaly type (unsupervised). Compares DEAN-TS with the best observed performance.

C.3 Anomaly Detection Performance by Dimensionality and Contamination

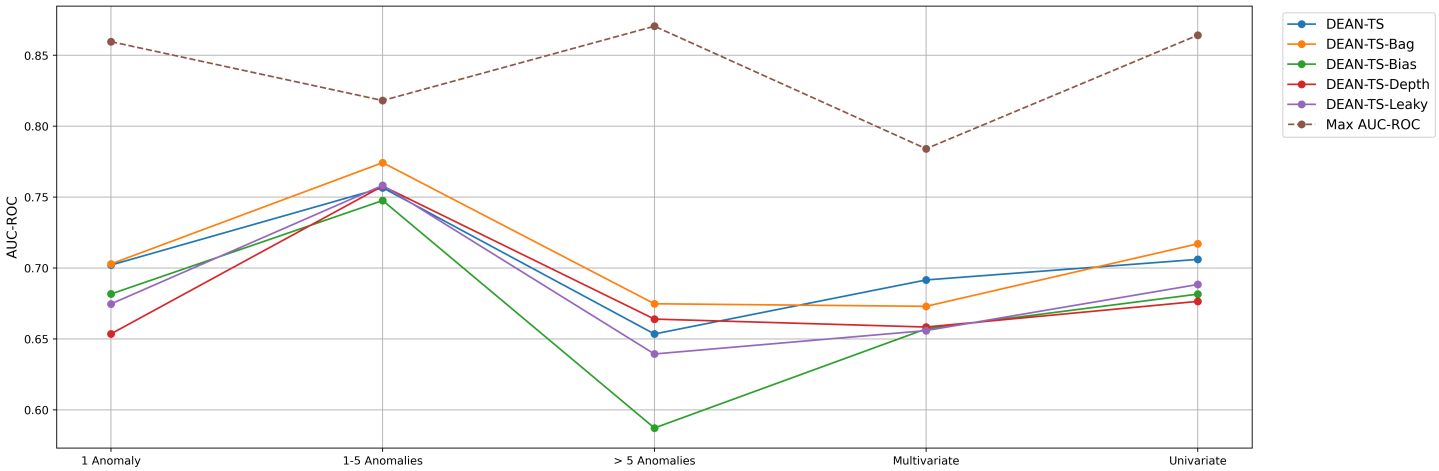


Figure A3.5: Average performance by dimensionality and contamination (semi-supervised). Compares DEAN-TS with the best observed performance.

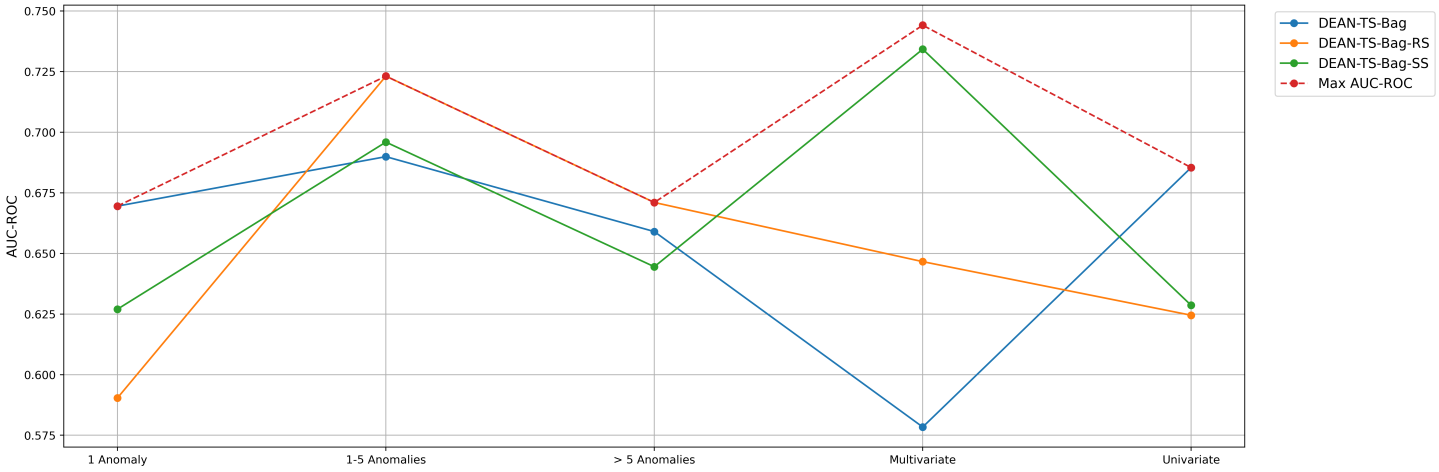


Figure A3.6: Average performance by dimensionality and contamination (unsupervised). Compares DEAN-TS with the best observed performance.

Eidesstattliche Versicherung

(Affidavit)

Katzke, Tim

Name, Vorname
(surname, first name)

168160

Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

■ Masterarbeit
(Master's thesis)

Titel
(Title)

DEAN-TS: Deep Ensemble Anomaly Detection for Time Series

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen- und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Oberhausen 10.07.2023

Ort, Datum
(place, date)

Tim Katzke

Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Oberhausen 10.07.2023

Ort, Datum
(place, date)

Tim Katzke

Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**