Explainable Anomaly Detection on High-Dimensional Time Series Data

Bijan Rad, Fei Song, Vincent Jacob, Yanlei Diao
École Polytechnique, Palaiseau, France
bijan.rad,fei.song,vincent.jacob,yanlei.diao@polytechnique.edu

ABSTRACT

As enterprise information systems are collecting event streams from various sources, the ability of a system to automatically detect anomalous events and further provide human-readable explanations is of paramount importance. In this paper, we present an approach to *integrated anomaly detection (AD) and explanation discovery (ED)*, which is able to leverage state-of-the-art Deep Learning (DL) techniques for anomaly detection, while being able to recover human-readable explanations for detected anomalies. At the core of the framework is a new human-interpretable dimensionality reduction (HIDR) method that not only reduces the dimensionality of the data, but also maintains a meaningful mapping from the original features to the transformed low-dimensional features. Such transformed features can be fed into any DL technique designed for anomaly detection, and the feature mapping will be used to recover human-readable explanations through a suite of new feature selection and explanation discovery methods. Evaluation using a recent explainable anomaly detection benchmark demonstrates the efficiency and effectiveness of HIDR for AD, and the result that while all three recent ED techniques failed to generate quality explanations on high-dimensional data, our HIDR-based ED framework can enable them to generate explanations with dramatic improvements in the quality of explanations and computational efficiency.

CCS CONCEPTS

• **Computing methodologies** → **Anomaly detection**; **Feature selection**; **Neural networks**; • **Information systems** → **Data stream mining**.

KEYWORDS

anomaly detection, explanation discovery, time series data analysis, dimensionality reduction, neural networks

ACM Reference Format:
Bijan Rad, Fei Song, Vincent Jacob, Yanlei Diao. 2021. Explainable Anomaly Detection on High-Dimensional Time Series Data. In *The 15th ACM International Conference on Distributed and Event-based Systems (DEBS '21), June 28-July 2, 2021, Virtual Event, Italy.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3465480.3468292

DEBS '21, June 28-July 2, 2021, Virtual Event, Italy

ACM ISBN 978-1-4503-8555-8/21/06.
https://doi.org/10.1145/3465480.3468292

1 INTRODUCTION

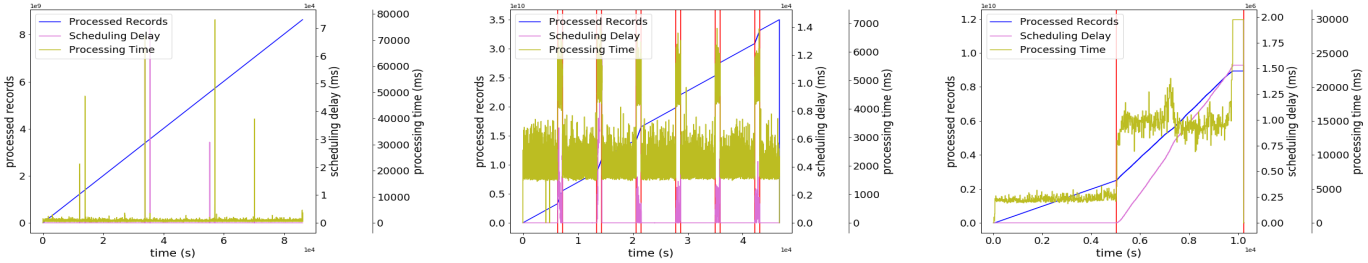
Enterprise information systems are collecting large amounts of event log data from various sources such as databases, transaction logs, audit trails, system monitors, and application monitors. Timely analysis of such event log data has become critical to business operations such as system health monitoring, application performance monitoring, and user behavior analysis.

As a concrete example, some of the largest E-commerce platforms are running Spark jobs on petabytes of data each day to analyze customer purchase patterns, target offers, and enhance customer experiences [32]. Since results of these jobs affect the immediate business operations for inventory management, sales strategies, etc., they are often specified with deadlines. Anomalies that occur in job execution would prevent analytical jobs from meeting their deadlines and hence cause disruption to critical operations on those E-commerce platforms. Therefore, there is a growing demand to collect event logs from the Spark applications as well as the underlying OS during job execution, and have a software tool to analyze these event logs in a timely manner in order to detect performance issues and enable corrective actions, such as allocating more resources, to meet application deadlines.

Motivated by the above use case, as well as similar use cases in other application domains, *our work aims to design a data analytics system that can automatically detect anomalies from high-volume event log data and further provide human-readable explanations for such events*, hence enabling preventive or corrective actions.

In this work, we consider anomalies as the patterns in data that deviate from expected behaviors [9]. An anomaly can be detected by an automatic procedure, for which existing techniques include statistical, SVM, and clustering based methods [5, 9, 14], as well as recent Deep Learning based methods [8]. However, there is one key component missing in all these techniques: finding the *best explanation* for the anomalies detected, or more precisely, a human-readable formula offering useful information about what has led to the anomaly. Without a good explanation, anomaly detection is only of limited use: the end user knows that something anomalous has just happened, but has little understanding of how it has arisen, how to react to the situation, and how to avoid it in the future.

To illustrate the importance of generating explanations for detected anomalies, Figure 1 shows three example recordings of Spark streaming application execution provided by the Exathlon anomaly detection benchmark [17] (described in more detail in the experimental section). Figure 1(a) shows the time series of three metrics that engineers often monitor in the Spark UI to check if their Spark streaming applications are making progress: (a) *scheduling delay* captures the delay between the scheduling of a task and the start of its processing; (b) *processing time* is the time taken to process



(a) Normal execution of a job (b) Job execution under bursty input (c) Job execution under bursty input until crash

Figure 1: Illustration of anomalies in Spark job execution. A pair of red vertical bars represent the interval of an anomaly

a batch of data from the streaming input; (c) *the number of processed records* in current execution. There are no anomalies in this job. However, the times series data already exhibits a great deal of variability: both scheduling delay and processing time can rise high beyond a normal range of values, which are likely to be caused by other system operations such as checkpointing in Spark or CPU usage by a DataNode in HDFS.

On the other day, the three metrics look somewhat different during job execution, as shown in Figure 1(b). Even if an anomaly detection tool could flag an alert for each interval in which the batch processing time and scheduling delay spike up, it is difficult for the user to figure out what is going on: "*What is happening with my submitted job*?" "*Is the phenomenon caused by the bugs in the code or some system anomalies*?" "*Should I wait for the job to complete or re-submit it*?" "*What should I do to bring the job progress back to normal*?" It turns out that the spikes are due to bursty input of streaming data – there are periods when the input rate rises significantly higher than its normal values, e.g., due to customer response to promotions or increased requests for a service. If the user could not understand the reason for the anomaly, it is hard to take actions to remedy the situation. Figure 1(c) further shows that if no actions are taken to allocate more resources to this application, an extended period of bursty input can push the application to crash, as shown by the number of processed records reset to zero.

Challenges. Designing an integrated analytics system for anomaly detection and explanation discovery raises a host of challenges.

First, these two topics have been addressed largely in isolation in the literature: while anomaly detection has been studied intensively in the data mining community [5, 9, 14], explanation discovery with the goal of human-readable formulas has recently received attention in the database community [25, 37, 41]. On the latter topic, recent works [25, 37] explain outliers for group-by aggregate queries and find a logical formula to describe a subset of tuples that contribute the most to the excessively high or low aggregate value of a specific group. It is hard to extend such work to explain anomalies detected by an arbitrary data mining algorithm. EXstream [41] assumes that the normal and abnormal time periods are already given by the user (treated as ground truth) and finds explanations to best distinguish the abnormal periods from the normal ones. It does not support automated anomaly detection if such ground truth is not available.

Second, existing integrated approaches cannot address complex anomalies that are of contextual or collective types [9] and need to be detected on high-dimensional time series data. The best known integrated system is MacroBase [3], a data analytics engine that helps the user prioritize attention over data streams, and offers modules for both outlier detection and explanation discovery. However, it performs outlier detection by a density-based method called MAD. While being simple and robust, MAD is suitable only for detecting point outliers, not contextual or collective outliers [9] which are often related to time series and sequence data.

Third, a recent position paper [31] argues for a new system that can harness the power of Deep Learning for detecting complex anomaly types, while at the same time can recover human-readable explanations for detected anomalies. In particular, Deep Neural Networks (DNNs) have demonstrated capabilities for handling high-dimensional time series data and detecting both contextual and collective anomalies [8]. However, DNNs are known to be hard to interpret, whose fully connected layers in a deep architecture essentially make the model a blackbox. How to obtain human-readable explanations while leveraging the power of DNNs for anomaly detection remains an unsolved issue.

Contributions. In this paper, we present an approach to *integrated anomaly detection and explanation discovery* that allows us to leverage state-of-the-art Deep Learning techniques for anomaly detection, while being able to recover human-readable explanations for detected anomalies. At the core of the framework is a new human-interpretable dimensionality reduction method that not only reduces the dimensionality of the data, but also maintains a meaningful mapping from the original features to the transformed low-dimensional features. Such transformed features can be fed into any DNN architecture designed for anomaly detection. Once an anomaly is flagged, we can leverage the feature mapping structure to recover human-readable explanations. More specifically, our contributions include:

1. *Human-Interpretable Dimensionality Reduction (HIDR).* We develop a human-interpretable dimensionality reduction (HIDR) technique with two distinct features: First, it reduces dimensionality by capturing statistical correlation among the features in the input dataset. Given the fact that many features in real-world event log data are correlated, HIDR clusters features based on their statistical correlation, and for each cluster of features, finds a new transformed feature through training an autoencoder that is able to reproduce all the correlated features in the cluster. Such transformed features can be fed into any DNN-based anomaly detection methods, such as Autoencoder-based [2, 39] and LSTM-based [6, 22] methods. Second, HIDR maintains an explicit mapping between original features and their corresponding transformed features, which will help us recover human-readable explanations using the original features.

2. *HIDR Feature Selection*. Since the transformed features are not interpretable, we further develop a HIDR-based feature selection

(FS) method to build a *filtered interpretable feature space* by leveraging the encoder gradient information from the cluster autoencoders trained in HIDR and selecting a subset from the original feature space for the ED task. Our work also customizes the cluster autoencoders to provide a theoretical guarantee that such feature selection enables consistent explanations to be discovered later.

3. *Human-Interpretable Low-dimensional Explanation Discovery (HILED).* Once an anomaly is flagged by the detection algorithm, our work returns an interpretable explanation using one of two approaches. First, we can run the feature selection to return a filtered interpretable feature space, as described above, and then any existing ED method on this space to return an explanation. Alternatively, we also provide a two-step approach that repeatedly calls existing ED methods to discover low and high-level explanations, optionally combined with our FS method, to further improve the quality of returned explanations.

4. *Evaluation.* Evaluation using a recent explainable anomaly detection benchmark [17] show the following results: (1) HIDR outperforms other dimensionality reduction methods in the size of reduced feature space, as well as in the accuracy of anomaly detection (AD) results. (2) None of the three state-of-the-art ED methods [3, 23, 41] can handle high-dimensional time series data. Using our HIDR Feature Selection method, these methods can generate explanations with dramatic improvements in conciseness and consistency of the explanations, as well as in computational efficiency. (3) Further, our HILED approach provides a flexible framework for trying different ways to combine an ED method with our gradient-based feature selection method, offering an opportunity for the user to fine-tune the performance to achieve better explanations.

2 RELATED WORK

Explaining outliers in SQL query results. Scorpion [37] explains outliers in group-by aggregate queries. Users annotate outliers on the results of group-by queries, and Scorpion then searches for predicates that remove these outliers while minimally affecting the legitimate answers. This method does not suit our problem because it works only for group-by aggregation queries and searches through various subsets of the tuples that were used to compute the query answers. Recent work [25] extends it to support richer and insightful explanations through some pre-computations enabling interactive explanation discovery. This work assumes a set of explanation templates given by the user and requires pre-computations in a given database. Neither of these assumptions fit our problem setting of explainable anomaly detection in incoming event log data, and it is hard to extend such a work to explain anomalies detected by an arbitrary machine learning algorithm. Given a multi-dimensional dataset, recent work [12] constructs an explanation table and finds patterns that affect a binary value of each tuple. This problem is different from ours because it tries to summarize existing data in a database with statistical information, without considering time series data, nor detecting and explaining anomalies in newly arriving time series data. Some recent industrial efforts were made towards time series anomaly explanation and root cause analysis in DB systems [18, 21]. These approaches however require a variety of inputs from the user, e.g., causal hypotheses [18], or root cause labels [21], while our work focuses on automatically learning for explainable anomaly detection.

Explaining outliers in stream processing. EXsteam [41] assumes that the normal and abnormal time periods are already given by the user (treated as ground truth) and finds explanations to best distinguish the abnormal periods from the normal ones. Some of its key techniques require user-labeled data, which is not available in our problem setting. The MacroBase [3] analytics engine is designed to help the user prioritize attention over data streams, with modules for both outlier detection and explanation discovery. However, it performs outlier detection by a simple density-based method called MAD. While being simple and robust, MAD is suitable only for detecting point outliers, not contextual or collective outliers [9] which are often related to time series and sequence data. For explanation discovery, MacroBase also encounters a scalability issue regarding the number of features in time series data. We will evaluate both of these techniques in our experimental study.

Explaining outputs in iterative analytics. Recent work [10] focuses on tracking, maintaining, and querying lineage and "how" provenance in the context of arbitrary iterative data flows. It aims to create a set of recursively defined rules that determine which records in a data-parallel computation inputs, intermediate records, and outputs require explanation. It allows one to identify when (i.e., the points in the computation) and how a data collection changes, and provides explanations for only these few changes.

Interpretable machine learning. In the ML community, there has been relevant work on interpretable models. Several projects have explored methods for obtaining interpretable classifiers based on perturbing the classifier inputs and observing the response [19, 23, 24, 33]. Among them, the most relevant work [23] proposed a model-agnostic approach, called LIME, that uses sparse linear models as explanations of any classifier. In particular, LIME explains a specific prediction of any classifier by approximating it locally with a visually interpretable linear model, and explains the overall model by selecting a set of representative instances with explanations. We evaluate LIME in our experimental study and demonstrate that it has a scalability issue for large datasets. Anchors [24] improved upon LIME by replacing a linear model with a logical rule that explains a single data instance and offers better coverage of data points in the local neighborhood, but it does not support time series data like in our work. SHAP scores [20] and RESP scores [4] are also instance-level explanations that assign a numerical score to each feature, representing their importance in the outcome. But these methods are computationally too expensive to suit data stream processing. Other recent works develop decision trees [13, 38] and decision sets [19] as explainable models. While decision trees are visually interpretable, their construction algorithm often fails for two main reasons [30]: (1) in the presence of class imbalance, such as in anomaly detection where anomalies are rare, the decision trees can simply classify all instances to the majority class; (2) the decision trees returned by the algorithm might be very large, making them hard to read and understand by a human.

Anomaly detection. Anomaly detection or outlier detection [9, 14, 35] has been studied intensively in the data mining community. Researchers are still contributing actively to this area to achieve more efficient detection algorithms [7, 16]. Common anomaly detection techniques (e.g., [7, 9, 14, 16, 35]) take one of two main approaches. In the first approach, a model is trained to perform a prediction task on data assumed mostly normal. An outlier is then defined as

an incoming data point for which the model makes a significant error. The other approach relies on distance functions [35], where outliers are defined as points that lie far from most of the others. A recent effort to handle high-dimensional data and complex types of anomalies was made through exploring new Deep Learning based techniques [8, 22, 26, 39]. However, all of the above approaches only report outliers, without the reasons (explanations) why they occurred. Our work aims to embrace DL-based anomaly detection methods while recovering human-readable explanations.

3 SYSTEM OVERVIEW

In this section, we present an overview of our EXplainable Anomaly Detection system, referred to as EXAD. EXAD analyzes incoming data streams, from which it flags some data windows as anomalous and generates an explanation for each detected anomaly.

Our system design is based on a *two-pass approach* to supporting *anomaly detection* (AD) and *explanation discovery* (ED) in the same stream analytics system. The approach is motivated by the observation that though closely related, AD and ED often differ in the nature of computation (e.g., the optimization objective), and it is hard to achieve both in a single procedure. Prior work [41] showed evidence when logistic regression is used to perform both AD and ED. While the regression model is a good predictive model for some anomalous events, it does not serve as a good explanation. For instance, the learned model assigns non-zero weights to 30 out of 345 input features, and it is hard for the human to understand an explanation with 30 features. Through manual exploration, the domain expert finally selected two features to construct an explanation; however, these two features are ranked low (23 and 24 out of 30) in the learned model. This indicates that the ED method needs to include a penalty term in the optimization objective to favor small explanations, while such a penalty term can often decrease accuracy of an AD model.

Therefore, our two-pass approach is designed to handle AD and ED in two different passes of the data. In the forward pass, the live data streams are used to drive anomaly detection and also archived for further analysis. The detected anomalies will be delivered immediately to the explanation discovery module. Then in the backward pass, explanation discovery runs on both the archived raw streams and feature sets created in the forward pass. Once the explanation is found, it is delivered to the user with only a slight delay.

Such decoupling of AD and ED functionalities allows EXAD to have an open architecture to embrace new AD and ED methods as they are proposed in the literature. There are many AD methods including recent Deep Neural Networks (DNNs) based ones [8] for detecting complex anomaly types, as well as multiple ED methods [3, 23, 41]. However, there are still technical challenges we face in the design of EXAD: (1) **AD/ED integration**: We still need to link these two models to be able to explain the results of the complex (e.g., DNN-based) anomaly detection model in a human-readable form, while DNN models are known to be hard to interpret. (2) **High-dimensionality**: The problem is further compounded by the fact that event streams are often in the form of high-dimensional time series data. While DNN-based AD methods have the ability to transform high-dimensional data into low-dimensional embeddings to enable anomaly detection, such embeddings are not interpretable. Hence, the ED methods cannot build explanations on these low-dimensional embeddings, but instead have to deal with the original high-dimensional data. As our evaluation will show, most ED methods [3, 23, 41] fail given 1000's of features in input data.

To address the above issues, EXAD employs a set of novel techniques, as shown in Figure 2. At the core of our system is a *human-interpretable dimensionality reduction* (HIDR) method to decrease the high dimensionality of time series data and to draw connections between the AD and ED models.

In *offline processing*, HIDR (a) clusters input features based on their statistical correlation, and (b) for each cluster of features, finds a single *transformed feature* through training an autoencoder (AE) that is able to reproduce all the correlated features in the cluster. The output of such offline processing includes a set of AEs, one for each feature cluster, and an explicit mapping between original features and their corresponding transformed features, which will be used to recover human-readable explanations using the original features.

In *online processing*, arriving data streams are transformed from the original feature space to the low-dimensional transformed feature space, by running the data through the encoder of each cluster AE. For the AD task, the AD model will run on the HIDR reduced space, which can greatly decrease its training time and online inference time. In order to get good detection results, however, this gain in efficiency must come with the transformed features from HIDR carrying sufficient information for anomaly detection, which our experimental study will verify.

Once an anomaly is flagged by the AD method, our work provides a new ED framework for generating explanations using the original features – the original feature space is interpretable to the human user whereas the transformed feature space is not. This is achieved via a suite of new techniques: (a) Our ED framework provides a HIDR-based feature selection (FS) method to build a *filtered interpretable feature space* by leveraging the encoder gradient information from the cluster AEs and selecting a subset from the original feature space. Then this space can be used by any existing ED task to return an explanation. (b) We further provide a two-step approach that repeatedly calls existing ED methods to discover low-level and high-level explanations, optionally combined with our FS method, to further improve the quality of returned explanations.

Notation in this paper. Let $\boldsymbol{x}^{(t)} \in \mathbb{R}^m$ denote each original data item in a time series, and $\boldsymbol{z}^{(t)} \in \mathbb{R}^d$ denote the transformed item. We have $d << m$. Moreover, let F_i denote the id of the i-th feature in each original data item $\boldsymbol{x}^{(t)}$, and V_j denote the id of the j-th feature in $\boldsymbol{z}^{(t)}$. Then we maintain explicitly the mapping $\Phi : \mathbb{R}^m \mapsto \mathbb{R}^d$ such that the features $\{F_i\}$ in the same cluster are mapped to the same embedding V_j. In addition, we use $x_i^{(t)}$ and $z_j^{(t)}$ to represent the value of the i-th feature in $\boldsymbol{x}^{(t)}$ and the j-th feature in $\boldsymbol{z}^{(t)}$, respectively.

3.1 Requirements and Survey of Related Work

Since real-world time series data is often of high dimensionality (e.g., including 100's to 1000's of features), the first requirement for effective explainable anomaly detection is dimensionality reduction. However, we have to generate explanations using the original features for the human user, which requires recovering the original features from the transformed space. This motivates the concept of a human-interpretable dimensionality reduction (HIDR) method.

More specifically, we have the following requirements on a dimensionality reduction method that can enable both anomaly detection

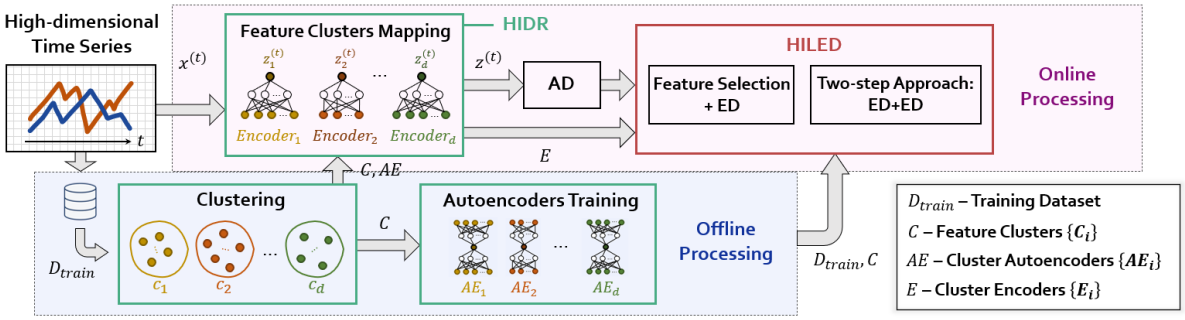


Figure 2: An integrated system for anomaly detection and explanation discovery

(AD) and explanation discovery (ED) on high-dimensional time series data: (1) *AD Efficiency*: whether AD is performed in a low-dimensional embedding space such that the offline training time and online inference time on data streams are minimized; (2) *AD Accuracy*: whether AD is provided with rich information in such low-dimensional embeddings such that the anomaly detection accuracy is maximized; (3) *ED Efficiency*: whether ED can be performed efficiently, that is, as soon as an anomaly is flagged on incoming data, an explanation is returned shortly to enable human inspection and potentially corrective actions. (4) *ED Interpretability*: whether the explanation returned by the ED method is built over compact, interpretable features that the human can easily understand.

We next survey the most relevant work on explainable anomaly detection on high-dimensional time series data and examine their performance with respect to the above requirements. Table 1 summarizes the analyses.

Dimensionality Reduction Methods: The first three rows of the table list common dimensionality reduction methods, including PCA [27], kernel PCA [27] and Factor Analysis (FA) [28]. They are known to be able to reduce high-dimensional data into low-dimensional representations (through linear or non-linear transformations), which can be further fed into any AD method (marked as AD*) for anomaly detection. The issue, however, is that these low-dimensional representations are not interpretable, and hence running an ED method on them will not return human-readable explanations.

Explanation Discovery Methods: The next three rows list three state-of-the-art ED methods, which we shall evaluate thoroughly in our performance study. EXstream [41] and LIME [23] were designed for the ED functionality only, and hence the AD functionality does not apply. For ED, EXstream is shown to be quite fast on high-dimensional data, but the interpretability is poor due to the lack of stability (i.e., the explanation for a particular instance changes a lot under just small perturbation of the data) or lack of consistency (i.e., the explanations of a set of anomalies that are known to have the same anomaly type do not agree with each other). This is largely due to the fact that high-dimensional time series data contains many features that are incidentally correlated with an anomaly period and the high dimensionality makes it hard for EXstream to discern the truly relevant ones. LIME [23] suffers in efficiency and cannot even run on the high-dimensional data (1000's of features) in our performance study. Macrobase [3] has both the AD and ED methods, which are disjoint methods. While the AD method of Macrobase is efficient but works only for point anomalies, its ED method lacks efficiency and also fails to run on our high-dimensional dataset.

	AD		ED	
	Efficiency	*Accuracy*	*Efficiency*	*Interpret.*
1) PCA [27] + AD* + ED*	Yes	Decent	Maybe	No
2) kPCA [27] + AD* + ED*	Yes	Variable	Maybe	No
3) FA [28] + AD* + ED*	Yes	Variable	Maybe	No
4) ED: EXstream [41]	–	–	Maybe	Poor
5) ED: LIME [23]	–	–	No	Poor
6) AD+ED: Macrobase [3]	Yes	Poor	No	Poor
7) HIDR + AD* + HILED	Yes	Good	Yes	Good

Table 1: Requirements on explainable anomaly detection on high-dimensional time series data and a survey of related work

The new methods provided in EXAD, including Human- Interpretable Dimensionality Reduction (HIDR) and Human-Interpretable Low-dimensional Explanation Discovery (HILED), finally enable both efficiency and accuracy in AD, as well as efficiency and interpretability in ED, as we shall show in our evaluation.

4 HIDR DIMENSIONALITY REDUCTION

To enable dimensionality reduction (DR) of input data into low-dimensional space while being able to explain the detection results, for each feature in the transformed low-dimensional space, we would like to understand which subset of input features and which method were used for its generation. This a key factor in the explanation generation phase. The current state of the art does not provide us with such a method. This motivated us to design a new method that is custom-made for this context.

HIDR Principle. A good method for performing dimensionality reduction on a data stream is to train an autoencoder [15], which performs a non-linear transformation of its input data into low-dimensional vectors via its encoder and then reconstructs the input via its decoder, with the goal to match the input and decoder output. In our context, we can apply its encoder to the data stream to reduce dimensionality. However, since the explanation discovery (ED) methods tend to analyze features in the low-dimensional space separately, there should be no information overlap between them. To solve this issue, our solution is to first perform correlation clustering on the original features and then train a separate autoencoder for each cluster having a one-dimensional representation.

More formally, our rationale for using an autoencoder is that each feature in our data is considered as a random variable. We can exploit the high correlations between the variables by regrouping them into correlation clusters and considering them as groups for our models.

Algorithm 1 HIDR: offline training

Require: Training dataset: $\{\boldsymbol{x}^{(t)}\}$
Require: Fearure set: $F = \{F_1, \ldots, F_m\}$
Require: Hierarchical clustering threshold: t
1: $M \leftarrow$ correlation_matrix($\{\boldsymbol{x}^{(t)}\}$), M is the pairwise correlation matrix
2: $C = \{C_1, \ldots, C_d\} \leftarrow$ form_cluster(M, t). $\forall i, j\ C_i \cap C_j = \emptyset$, and $\cup_{i=1}^{d} C_i = F$
3: **for** each $C_i \in C$ **do**
4: $E_i \leftarrow$ train_autoencoder($\{\boldsymbol{x}_{C_i}^{(t)}\}$), the output layer of encoder E_i is one-dimensional
5: **end for**
6: **return** $C = \{C_1, \ldots, C_d\}$, $E = \{E_1, \ldots, E_d\}$

Algorithm 2 HIDR: online transformation

Require: Dataset to be transformed: $\{\boldsymbol{x}^{(t)}\}$
Require: Clustering groups: $C = \{C_1, \ldots, C_d\}$
Require: Encoders: $E = \{E_1, \ldots, E_d\}$
1: **for** each $C_i \in C$ **do**
2: $\{z_i^{(t)}\} \leftarrow E_i(\{\boldsymbol{x}_{C_i}^{(t)}\})$
3: **end for**
4: **return** $\{\boldsymbol{z}^{(t)}\} = \{[z_1^{(t)}, \ldots, z_d^{(t)}]\}$

For each cluster, we can find a single underlying variable that can generate all the features of that cluster, due to their high pairwise correlation. This could be achieved using an autoencoder trained to minimize its reconstruction error, with its encoder being the mapping from each cluster of correlated features to the underlying variable. In contrast, a standard clustering method such as computing the centroid on its own is not general enough since it considers all features to be equally important, which is generally not the case.

Algorithm. We divide the HIDR method into offline training as in Algorithm 1, and online transformation as in Algorithm 2.

In the offline training phase (Algorithm 1), the algorithm takes a training dataset as input. It first computes the pairwise Pearson correlation coefficients between the features, thereby producing the correlation matrix. To reduce redundancy in this dataset, the algorithm seeks to regroup these features by clustering them. After the correlation clustering, the algorithm moves onto dimensionality reduction using autoencoders. For each cluster, the algorithm trains a feedforward autoencoder [15] taking the features of that cluster as input and learning their one-dimensional representation through minimizing its reconstruction error. The final output of this phase includes the feature clusters, $\{C_1, \ldots, C_d\}$, and the encoder for each cluster, $\{E_1, \ldots, E_d\}$.

In the online transformation phase (Algorithm 2), the algorithm takes an incoming dataset as input. Using the encoders trained from the offline phase, $\{E_1, \ldots, E_d\}$, it can transform each data point, $\boldsymbol{x}^{(t)} \in \mathbb{R}^m$, from the original feature space to the transformed feature space $\mathbb{R}^d$, where $d << m$, thereby generating a low-dimensional vector, $\boldsymbol{z}^{(t)} = [z_1^{(t)}, \ldots, z_d^{(t)}]$ in the transformed space.

Implementation Details. Throughout EXAD, we use one large reference normal trace for the correlation clustering, and all the normal traces for training the autoencoders. In the clustering step, we do not specify the number of clusters to the method; otherwise, it would potentially force some features into clusters where they do not belong. The method that is best fit to this context is hierarchical clustering [1]. By applying this method to our high-dimensional dataset with 1,055 features (to be detailed in our evaluation section), we obtained 13 clusters which in turn will reduce the dimension of our data from 1,055 to 13.

Algorithm 3 HIDR Feature Selection

Require: Dataset for the gradient calculation: $\{\boldsymbol{x}^{(t)}\}$
Require: Clustering groups: $C = \{C_1, \ldots, C_d\}$
Require: Encoders: $E = \{E_1, \ldots, E_d\}$
Require: Threshold for gradient selection: t
1: FS $\leftarrow \{\}$, FS is the set of important features to be selected
2: **for** each $C_i \in C$ **do**
3: $G = \{G_1, \ldots, G_{|C_i|}\} \leftarrow$ cal_gradient(E_i, $\{\boldsymbol{x}_{C_i}^{(t)}\}$)
4: **for** each $j \in \{1, \ldots, |C_i|\}$ **do**
5: **if** $G_j > t$ **then**
6: add index $C_i(j)$ to FS
7: **end if**
8: **end for**
9: **end for**
10: **return** FS

Advantages over other DR methods. One of the advantages of the HIDR method is that we know which features were combined in order to create each specific variable in the transformed space. This is not the case for methods such as PCA or kernel PCA. This is very important for the explanation phase since we must present explanations using the original features to human users.

The most similar work is the Factor Analysis method. In Factor Analysiswe assume that the observations on the original feature space are a linear transformation of independent latent features (factors) in a space of a lower dimension plus some added Gaussian noise. Without loss of generality, the noise as well as each factor are assumed to follow a Gaussian distribution with zero mean and unit covariance. As such, it tries to reduce the dimensionality and pairwise correlation of the transformed space. However, one cannot separate the latent factors during explanation discovery and more specifically, cannot find the relationship between a latent factor and the original features. Therefore it can not be used for the ED task.

Another method for interpretable dimensionality reduction is Isomap [34]. It is a non-linear dimensionality reduction algorithm which considers the intrinsic geometry of the data cloud. Although it is geometrically interpretable, it fails to provide the human users with a simple relevant explanation.

5 HIDR FEATURE SELECTION

HIDR relies on the autoencoders to reduce the dimensionality. This results in an irreversible feature transformation. However, in some tasks such as explanation discovery (ED), it is desirable that the features to be used carry the semantic meanings. The feature set generated by HIDR does not satisfy this requirement. Motivated by this observation, and leveraging the Encoder Gradient information, we further develop a feature selection method based on HIDR, and call it HIDR Feature Selection. This method selects a subset of the original feature space, achieving the dimensionality reduction and keeping the semantic meaning of each feature at the same time. We summarize this method in Algorithm 3 and discuss it more below.

Encoder Gradients. Inspired by LIME [23] and Saliency maps [29], our solution to feature selection is based on computing the gradient of each cluster encoder with respect to its input features.

Mathematically, every encoder E is a mapping $e\colon \mathbb{R}^n \to \mathbb{R}$ where n is the number of features of its corresponding cluster. It can be seen as the function mapping from an input vector $\boldsymbol{x}^{(t)}$ to its representation feature $\boldsymbol{z}^{(t)}$. In addition, for each input vector $\boldsymbol{x}^{(t)}$, the gradient $\nabla e(\boldsymbol{x}^{(t)})$ gives the importance of each feature F_i in $\boldsymbol{x}^{(t)}$ given the mapping e. This indicates that if we use the reconstruction error to train the autoencoders, then we can extract the most important features from a correlation cluster by choosing the ones with higher corresponding absolute values in the gradient.

In theory, for this extraction, we could use any activation function for the autoencoder. However, we here want to choose an activation function that guarantees the *consistency* property when deriving the feature importance. The consistency of an explanation is an important property we must achieve. Intuitively, it means that the explanations of similar anomalous datasets should be similar. To ensure this property, we have to impose a constraint on the autoencoders. Mathematically speaking, to ensure explanation consistency, the encoder mapping e must be a *uniformly continuous function* having a *bounded uniformly continuous gradient*.

Next, we first define consistency within the encoder context. Then, we introduce the proposition stating the conditions under which this consistency is guaranteed. After the proof of this proposition, we discuss the choice we made for our activation function.

Consistency Property: Let E be an encoder, and $e\colon \mathbb{R}^n \to \mathbb{R}$ the corresponding mapping. Consistency in this context means that similar input vectors $\boldsymbol{x}^{(t)}$ and $\boldsymbol{y}^{(t)}$ must have similar representations $e(\boldsymbol{x}^{(t)})$ and $e(\boldsymbol{y}^{(t)})$, as well as similar explanations (in our case, $\nabla e(\boldsymbol{x}^{(t)})$ and $\nabla e(\boldsymbol{y}^{(t)})$). Mathematically:

(1) $\|\boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)}\| \leq \eta \implies |e(\boldsymbol{x}^{(t)}) - e(\boldsymbol{y}^{(t)})| \leq \epsilon$
(2) $\|\boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)}\| \leq \eta \implies \|\nabla e(\boldsymbol{x}^{(t)}) - \nabla e(\boldsymbol{y}^{(t)})\| \leq \gamma$

where η, ϵ, γ are positive constants.

Proposition 5.1. If all of its activation functions are *uniformly continuous* (hereafter referred to as UC) and have *bounded uniformly continuous* derivatives (hereafter referred to BUC) , then a uniformly continuous function e with a uniformly continuous gradient can be learned by the neural network.

This proposition states that if all the activation functions of a neural network are both UC and BUC, then the **Consistency Property** is guaranteed. This proposition can be proved by induction on the number of layers of the network:

- **Base Case**: The first layer h_1 has the necessary properties.
- **Inductive Hypothesis**: Assume the layer h_n shares these properties.
- **Inductive step**: Show that every neuron in the layer h_{n+1} has a bounded uniform continuous (BUC) derivative with respect to any variable $\boldsymbol{x}_{F_i}^{(t)}$ in any given vector $\boldsymbol{x}^{(t)}$.

Now we show the proof for the inductive step. For the ease of notation, we drop the superscript t, and write $\boldsymbol{x}_{F_i}^{(t)}$ simply as $\boldsymbol{x}_k$.

Proof 5.1. We prove that any neuron in layer h_{n+1} has a BUC derivative with respect to any $\boldsymbol{x}_k$. If we denote the output of a given neuron as o, then this amounts to proving that $\frac{\partial o}{\partial \boldsymbol{x}_k}$ is BUC.

Inductive step: Let g be the activation function of a neuron in layer h_{n+1}. The activation value of this neuron then reads:

$$o = g\left(\sum_{i=1}^{l} w_i \cdot h_n^{(i)}\right)$$

Leading to the following partial derivative:

$$\frac{\partial o}{\partial \boldsymbol{x}_k} = \sum_{i=1}^{l} \frac{\partial o}{\partial h_n^{(i)}} \frac{\partial h_n^{(i)}}{\partial \boldsymbol{x}_k}$$

Which, by applying the chain rule, is equal to:

$$\frac{\partial o}{\partial \boldsymbol{x}_k} = \left(\sum_{i=1}^{l} \frac{\partial h_n^{(i)}}{\partial \boldsymbol{x}_k}\right) \cdot g'\left(\sum_{i=1}^{l} w_i \cdot h_n^{(i)}\right)$$

Where the righthand side is BUC as a linear combination, composition and product of BUC functions (from the Inductive Hypothesis, h_n is BUC), proving the result of the inductive step.

By induction, we conclude that the function e learned by a neural network having the specified properties is UC with a BUC gradient, which means that it gives consistent representations with consistent explanations, thus satisfying the **Consistency Property**.

Now the only issue left is the choice of the activation function for our autoencoders. To ensure the consistency property for the derived explanations, we saw that our activation function must be UC with a BUC gradient. For this reason, in this work, we chose to use the *ELU* (Exponential Linear Unit) activation function for our autoencoders instead of the more commonly used *ReLU* (Rectified Linear Unit) function. More specifically, ELU and ReLU take both the form of an identity function for non-negative inputs. For negative inputs, however, ELU changes smoothly and slowly as its output approaches a given negative parameter, while ReLU turns directly to zero, making a sharp turn. To conclude, having this architecture with ELU activation functions, we can expect the encoders gradients to give us consistent explanations in the required format.

6 LOW-DIMENSIONAL EXPLANATION DISCOVERY

As discussed above, the output of the HIDR Feature Selection method can be used directly as the input to any existing ED method. However, we noticed that this method does not fully leverage the clustering information. In HIDR dimensionality reduction, each cluster has been transformed to one variable, where the clustering information has been used. In contrast, in HIDR Feature Selection, the clustering/correlation information is used only in the training of the encoder, and then wasted. Motivated by this, we developed a two-step approach which is called HILED (Human-Interpretable Low-dimensional Explanation Discovery). It tries to leverage the clustering information as much as possible, and by introducing the use of an ED algorithm in both steps, it has the potential to generate high-quality explanations.

The key idea is as follows: First, we call HIDR to transform the dataset to a low-dimensional space. Then we call any existing ED method to generate an explanation. This is the first step of the HILED where the result is a subset of features in the reduced space, $V' \subseteq \{V_1, \ldots, V_d\}$. We call this subset the *low-level explanation*. Note that these features are not interpretable yet, but correspond to only a subset of clusters (hence simpler to deal with). Then we use

only the feature subset $\{F_i\}$ from each of the corresponding clusters, run an ED method again and generate the explanation using the original features, which is called a *high-level explanation*.

6.1 Leveraging Existing ED Methods

The first step of HILED is to call an ED method to select a subset of features in the transformed space to be the most important ones. Any existing ED algorithm that takes as input the dataset and ground truth (label) and outputs the important feature subset can be used in HILED. In this work, we experiment with three existing algorithms.

EXstream [41] is an explanation model specifically designed for monitoring high-volume event streams from enterprise information systems [41]. This model compares a reference normal period with an anomalous period (which is detected by the AD method of choice) by analyzing the segmentation of the values of the two periods. This way, it looks for an anomalous interval of the values of each feature; for a given feature, the more we can separate the normal values from the anomalous ones, the more important it becomes in explaining the detected anomaly.

The method starts by computing a single-feature reward for every feature and only keeps the ones with the highest scores. It then performs a correlation clustering to detect the pairs of correlated features and discards the feature with the lower rewards. This results in choosing a subset $V^* = \{V_1^*, \ldots, V_l^*\}$ from the original features V which have the most important contribution to the anomalous behavior. It generates explanations in the form of $(V_1^* \in S_1) \wedge \cdots \wedge (V_l^* \in S_l)$, where $S_i \subset \mathbb{R}$ is the range of anomalous values corresponding to V_i^* for all $i \in \{1, \ldots, l\}$.

LIME [23] was originally designed to derive explanations for a classifier's predictions, but also provides an interface for regression models and temporal data. Internally, LIME first leverages Lasso [11] to identify k important features, and then uses a customized loss function to learn a new linear model to approximate the original model's output locally. In our implementation, we used the RecurrentTabularExplainer interface provided by LIME, with $k = 5$.

MacroBase [3] proposes an explanation method called the MDP explanation operator. The algorithm and logic of this method were designed for categorical features (referred to as attributes). This operator takes as input a set of outliers O and inliers I, as well as two parameters called the minimum risk ratio, r, and minimum support, s. It first finds attributes with sufficient support in O and sufficiently high risk ratio in O and I, to remove the attributes that do not contribute to anomalous behaviors. It then runs frequent itemset mining (building an FP-Growth Tree) over O using only the previously found attributes to form the best attribute combination as the final explanation. This method was designed exclusively for explaining datasets consisting of categorical features, but not numerical features. In our work, we applied discretization to extract categorical features from our dataset.

6.2 The HILED Approach

We next explain how our HILED technique uses an existing ED method in a two-step approach, as shown in Algorithm 4.

Low-level Explanation. The input to the first step is the dataset $\{\boldsymbol{x}^{(t)}\}$ that was marked as anomalous by an AD method. Depending on the ED method used, a normal dataset might be required to serve as a reference. Then, our system extends $\{\boldsymbol{x}^{(t)}\}$ with a time window of the data directly preceding it, assumed to be normal as not flagged anomalous by the AD method. As Algorithm 4 shows, the low-level explanation discovery step applies HIDR online transformation (i.e., running $\{\boldsymbol{x}^{(t)}\}$ through the cluster encoders) to generate the features $\{\boldsymbol{z}^{(t)}\}$ in the reduced space. The output of the ED method is a selected set of important features, $V' = \{V_{i1}, \ldots, V_{im}\}$, in the reduced space.

Algorithm 4 HILED

Require: Dataset to be explained: $\{\boldsymbol{x}^{(t)}\}$
Require: Point-wise label information from the AD method: $\{l\}$, $l \in \{0, 1\}$
Require: Clustering groups: $C = \{C_1, \ldots, C_d\}$
Require: Encoders: $E = \{E_1, \ldots, E_d\}$
Require: Additional parameters p for the ED method
1: $\{\boldsymbol{z}^{(t)}\} \leftarrow$ HIDR_online_trans($\{\boldsymbol{x}^{(t)}\}$, C, E)
2: $V' = \{V_{i1}, \ldots, V_{im}\} \leftarrow$ ED($\{\boldsymbol{z}^{(t)}\}$, $\{l\}$, p). This is the end of the first step, low-level explanation
3: $F' = \cup_{j=1}^{m} C_{V_{ij}}$
4: [optional] $F' \leftarrow$ HIDR_feature_selection(F', C, E). This is the optional step to further reduce the dimensionality
5: $\{F_{i1}, \ldots, F_{in}\} \leftarrow$ ED($\{\boldsymbol{x}_{F'}^{(t)}\}$, $\{l\}$, p). This is the second step, high-level explanation
6: **return** $\{F_{i1}, \ldots, F_{in}\}$

By performing this step, we gain two advantages compared to running the ED method on the raw input data directly. First, we can fully leverage the clustering information. Second, it will reduce the complexity dramatically compared to using the raw feature space. For example, if we denote the number of features as m, and only consider the complexity terms regarding m, Macrobase is exponential in the worst case, $O(2^m)$; EXstream is $O(m^3)$, and LIME is also $O(m^3)$. Hence, reducing the number of features from $m = 1055$ to $d = 13$ will certainly bring enormous performance benefits.

Optional Step for Further Dimensionality Reduction. For the ED task, we need to use the original features, not the transformed features, to return interpretable explanations. A straightforward solution is as step 3 in Algorithm 4, which constructs a *filtered interpretable feature space* by merging the original features only from the clusters selected in the low-level explanation. However, this filtered feature space may still contain too many features for an ED algorithm to handle. For example, Macrobase cannot run on this filtered feature space (the number of features per cluster is $\frac{1055}{13}$ and the worst case complexity of Macrobase is exponential). In such a scenario, we need an optional step to further prune the interpretable feature space.

This optional step is step 4 in Algorithm 4. For each transformed feature $V_i \in V'$, we compute the gradient of the corresponding encoder ∇E_i and keep only the original features corresponding to the top values of the gradient. This way, we can find the most informative features within a cluster which can also have the highest contribution to the anomaly. More precisely, the low-level explanation gives us a subset of the reduced features that are causing the anomaly (V'). Also, for each $V_i \in V'$ we know the cluster of features C_i and the corresponding encoder E_i. Therefore, in order to find the original features that had the greatest effect on the anomalous behavior, we can analyze the effect of every feature in C_i on the behavior of E_i. Since the autoencoders are trained using the reconstruction error, finding the most important input variables of the encoder implies

finding the most important features in the correlation cluster. This step is equivalent to running HIDR Feature Selection on the clusters selected in the low-level explanation discovery.

High-Level Explanation. Given the filtered interpretable feature space (computed by step 3 and optionally, step 4), the next step is to discover the high-level explanation (step 5). This step will run an existing ED method on this filtered feature space and generate the final explanation. The interpretability comes with the usage of the original features, and the efficiency comes from using only a subset of these original features (step 1 and optionally, step 4).

7 EXPERIMENTAL RESULTS

In this section, we evaluate our HIDR and HILED techniques using a recent benchmark for explainable anomaly detection [17].

7.1 Experimental Setup

We implemented EXAD with three main modules:

1. Dimensionality reduction (DR) using (*i*) the HIDR method, (*ii*) PCA [27], a standard DR method, and (*iii*) Factor Analysis [28, 36], a more elaborate data reduction technique that works by modeling the observed data from theoretical latent factors.

2. Anomaly detection using an autoencoder (AE) network [15] along with an unsupervised threshold selection method. First, the AE is trained to accurately reconstruct time windows of normal data only. The outlier score of a test window is then defined as proportional to the reconstruction error by the AE, and the outlier score of a test record as the average outlier score of the windows it belongs to. The final threshold on the outlier scores, above which we flag a record as anomalous, is determined through the statistical modeling of a subset of the normal data, using the same methodology as in [17].

3. The HILED approach for explanation discovery, including (1) HIDR feature selection followed by an existing ED method; (2) two-level explanation discovery, denoted as ED + ED for low-level and high-level explanations, respectively, optionally with feature selection before the high-level explanation method is run. We have integrated a number of existing ED methods in EXAD for evaluation: (*i*) the gradient-based method described in the previous sections; (*ii*) EXstream [41], (*iii*) MacroBase [3], (*iv*) LIME [23].

Anomaly Detection Benchmark. We used the Exathlon anomaly detection benchmark [17]. This benchmark was designed for detecting anomalies in the execution of Spark streaming applications based on metrics collected from both the Spark UI (Monitoring and Instrumentation) interface and the underlying OS. The Exathlon dataset includes 93 *traces* and a *ground truth table*. The 93 traces, corresponding to 93 recurrent runs of Spark streaming applications, contain a total of 2,335,781 records, collected at the frequency of each second over 27 days. In their raw format, the records contain 2,283 metrics (features), including (*i*) 243 Spark driver metrics; (*ii*) 140 Spark executor metrics for each of 5 executor spots that can be used in a Spark application; (*iii*) 335 OS metrics for each of the 4 nodes of the cluster the Spark applications were run on. Hence, the benchmark presents high-dimensional time series data for analysis.

Among the 93 traces, 59 were left *undisturbed*, while 34 were *disturbed* by injecting external events of 6 different types: bursty input, bursty input until crash, stalled input, CPU contention, driver failure and executor failure (more details regarding the anomaly types can be found in [17]). Each disturbed trace includes periods of normal state as well as abnormal state corresponding to a given type of anomaly. The ground truth table records a total of 97 anomaly instances with their root cause events, in rows of the form *(app_id, trace_id, anomaly_type, root_cause_start, root_cause_end, extended_effect_start, extended_effect_end)*. Throughout this study, the anomalous intervals are considered as spanning from the start of the root cause to the end of the extended effect.

Feature Space	Size
Preprocessed original features	1055
PCA keeping 95% of the variance	557
HI Dimensionality Reduction	**13**

Table 2: The size of different feature spaces. Note that Factor Analysis is not included in the table because it must take the number of factors as input, but cannot automatically identify an appropriate number itself

Feature Preprocessing. Prior to using any of the techniques described, we applied some basic preprocessing to the raw features of the Exathlon dataset. This preprocessing involved dropping some unused metrics and performing some aggregation, leading us to having **1,055 features** as input, instead of the original 2,283 features.

System. All of our experiments were carried out on a server with 2 Intel Xeon Gold 6130 processors with 16 cores each, 768GB of memory, and 64 TB disks.

7.2 HI Dimensionality Reduction

In this section, we compare HIDR to other popular dimensionality reduction methods from the state of the art. We chose PCA [27] and Factor Analysis [28] as representatives. PCA is the most commonly used method for tabular data, while Factor Analysis is the closest method to our work: it aims to find underlying variables in the data with very little pairwise statistical correlation, which is one of the essential ideas behind HIDR. The purpose of this experiment is to examine how HIDR compares to PCA or Factor Analysis for dimensionality reduction as a means for anomaly detection.

Size Comparison. Given the very high dimensionality of our dataset, the first measure we consider is the number of features resulting from each method. We present these results in Table 2. From this table, we observe that applying HIDR results in a significantly smaller number of features than PCA keeping 95% of variance, with PCA leading to more than forty times as many features. For a fairer comparison to HIDR with respect to the input dimensionality, we also consider both PCA and Factor Analysis with their number of output features explicitly set to 13 in the following.

Accuracy of Anomaly Detection. We next compare the performance of our anomaly detection model on different feature spaces to evaluate the amount of information carried by each feature space.

Table 3 presents the AD results for this experiment, averaged across five different runs. *CUSTOM* refers to using the 19 manually constructed features of FS_{custom} in [17], that we treat as an upper-bound performance in our experiment. *NO DR* stands for "no dimensionality reduction" and *FA* for "Factor Analysis". For each run, the Autoencoder-based anomaly detection (AD) model was trained on the undisturbed traces of the Exathlon dataset and tested on its disturbed traces (*LS4* in [17]). The same AD model hyperparameters were used for each dimensionality reduction method. The AD performance is here reported as the "global point-based"

Method	F1-score	Precision	Recall
CUSTOM	0.59	0.50	0.72
NO DR	0.57	0.46	0.75
HIDR	0.58	0.48	0.74
PCA (95%)	0.57	0.49	0.70
PCA (13)	0.52	0.46	0.59
FA (13)	0.50	0.42	0.62

Table 3: Accuracy of Autoencoder-based anomaly detection on different feature spaces

Method	DR	SCORING	TOTAL
NO DR	N/A	20.5	20.5
HIDR	671.6	10.9	682.5
PCA (95%)	6.8	16.7	23.5
PCA (13)	4.6	11.3	15.8
FA (13)	5.1	11.0	16.1

Table 4: Inference speed of Autoencoder-based anomaly detection on different feature spaces in μs/record

F1-score, Precision and Recall, that is, considering all anomaly types the same and using the classical, point-based, definitions of Precision and Recall. For each run of the experiment, we evaluated the SD, MAD and IQR thresholding methods [40] fit on a subset of undisturbed traces that was not used for the AD model training/validation, each with the same 8 parameter combinations as in [17], and retained only the threshold resulting in the best F1-score.

For the HIDR method, the correlation matrix was computed on the largest undisturbed trace only, and the autoencoders were trained for 20 epochs per trace, to remove any biases arising from different traces having different lengths.

The main observations from Table 3 are: (1) The AD performance using the features produced by HIDR is comparable to the one obtained without dimensionality reduction and using PCA with 95% of kept variance, while also being close to the one obtained with the FS_{custom} feature set. On the other hand, HIDR achieves such accuracy using only 13 features, as opposed to 1,055 raw features, or 577 features based on PCA (95%). (2) When PCA and Factor Analysis are asked to use only 13 features, as in HIDR, there is an obvious drop in accuracy. HIDR significantly outperforms these methods that also used 13 output features in accuracy, while also being more interpretable. These results show that the best AD performance can be maintained using HIDR while working with a much more efficient and interpretable feature space.

Training Efficiency. Using HIDR, the average training epoch time for the AD model was reduced by a factor of around 6 compared to PCA (95%), and by a factor of 14 compared to not applying dimensionality reduction.

Online Inference. Table 4 reports the details of inference speed as live data arrives for the five considered methods, for both the dimensionality reduction (*DR*) and AD model computation (i.e., computing the outlier scores, labeled as *SCORING*) steps. These results show that, despite the HIDR inference being slower than the other regular dimensionality reduction methods, it is still able to process data records much faster than they arrive, with a total inference rate of around 1,500 records/sec, compared to the data arrival rate of one record/sec for each running Spark application of this benchmark dataset. This means that our inference speed is fast enough to handle around 1,500 Spark applications on a single CPU (assuming that there is enough network bandwidth to do so).

7.3 ED Evaluation Results and Discussion

In this section, we report the results of running existing ED algorithms within the HILED framework. We selected EXstream [41], MacroBase [3], and LIME [23] (tabular explanations and recurrent tabular explanations) as the explanation methods due to their popularity in the databases and machine learning communities.

Regarding evaluation metrics, we adopt the ones of the Exathlon benchmark [17], used in the context of the ED1 and ED2 setups of its *Explanation Discovery Functionality*. For the sake of completeness, we briefly describe these concepts here.

In the ED1 setup (*Local Explanation*), we evaluate the explanation of each anomaly instance individually. The explanation returned for an anomaly instance is a set of *important features*. (1) *Conciseness* is the length of each explanation (size of the feature set). (2) *Consistency* captures how stable the explanation is. It is computed by running the ED method multiple times where each run only uses a random selection of 80% of the data from the detected anomaly interval to generate the explanation. Multiple runs of the ED method return a set of explanations. We put all of these explanations into a combined feature set, and then compare how they agree with each other by calculating the entropy of the combined set (ideally, the fewer features in the set, the better). This entropy-based measurement is called consistency. (3) *Normalized Consistency*: However, the consistency metric depends also on the size of the explanation (i.e., conciseness). Hence we use one more measurement, called the normalized consistency, to eliminate the impact of the conciseness. The formula for it is $\frac{2^{consistency}}{conciseness}$, and its best value is 1.

In the ED2 setup (*Global Explanation*), we consider multiple anomaly instances that belong to the same anomaly type (based on the ground truth given for evaluation). The ED2 measurements capture how the explanations of the same anomaly type agree with each other. (1) *Conciseness* is defined as the average explanation size of these multiple anomalies. (2) *Consistency*: We put the explanations of multiple instances of the same anomaly type into a combined feature set, and then calculate the entropy of the combined set (again, the fewer features in the set, the better). (3) *Normalized Consistency* is defined as in ED1.

EXPT1: Generating Explanations From Raw Features. In the first experiment, we generate explanations from the original feature space using the three ED methods. For each method, we report in Table 5 the conciseness, consistency, and normalized consistency for ED1 and ED2, as well as the running time in second. Each row corresponds to a specific anomaly type (listed as T1-T6), and the final row is the average result across all anomaly types.

As we can see in Table 5, with 1,055 features in the raw input data, Macrobase and LIME cannot return any result within 24 hours. EXstream manages to generate explanations. However, the consistency performance is poor: the normalized consistency for ED2 is 10.38, far from the ideal value of 1. This is because high-dimensional time series data contains many features that are incidentally correlated with an anomaly period, and the high dimensionality makes it hard for EXstream to discern the truly relevant ones. The results

	EXstream							MB	LIME
	Concise		Consistency		Norm.Consis		Time (sec)		
	ED1	ED2	ED1	ED2	ED1	ED2	ED1/2		
T1	1.83	2.03	1.00	5.29	1.55	19.28	1.10		
T2	5.60	4.71	2.95	4.66	1.77	5.35	0.52	N/A	N/A
T3	3.92	5.42	2.24	5.58	2.17	8.83	1.05		
T4	3.54	4.25	2.22	6.00	2.07	15.06	0.61		
T5	11.46	5.71	3.52	5.32	1.92	7.00	0.42		
T6	6.25	6.25	3.15	5.40	1.92	6.77	0.55		
Ave	5.43	4.73	2.51	5.38	1.90	10.38	0.71	N/A	N/A

Table 5: Results of explanations returned by EXstream, MacroBase (MB), and LIME when running on raw input data (1,055 features)

here indicate that *none* of the three algorithms are able to generate quality explanations while handling the high dimensionality.

EXPT2: Generating Explanations From Low-Level Features: Using HIDR for Dimensionality Reduction. In this experiment, we generate explanations from the feature space of HIDR. For each cluster of HIDR, we use its encoder as a feature transformation tool, to generate one new feature per cluster. This way, for each data point in the original feature space, we transform it to a data point in a low-dimensional feature space. The dimensionality is significantly reduced; in this dataset, we reduce the number of features from 1,055 to 13. In the HILED terminology, the explanation built on this reduced feature space is called a *low-level explanation*.

In Table 6, we list all the results of the three ED methods for building low-level explanations. Here we can draw two main conclusions. First, LIME and Macrobase can now generate results, while previously they could not. Second, the performance of EXstream has been significantly improved.

We should emphasize that despite such an improvement, there is a significant drawback limiting the usage of low-level explanations: the low-level features do not carry any semantic meaning for a human user. However, we listed the results here for two purposes. First, we show the potential improvements assuming these low-level features can be understood by a human user – hence these improvements represent an "upper bound" on what an interpretable ED method can actually achieve. Second, and more importantly, we are going to use the low-level explanation as the intermediate step in the two-step HILED approach. Given our current evaluation results, we decide to use Macrobase as the ED method for the first step of HILED in the following experiments.

Also note that, from this point onward, we only use Macrobase and EXstream as the ED algorithms. We do not evaluate LIME, because it requires a pre-trained AE model on a fixed feature set. In the following experiments, it is very often that we need to vary the feature set according to the different input datasets, and was impractical to train AE models on different feature sets dynamically.

EXPT3: Generating Explanations using Gradient Information: Using HI as Feature Selection. As discussed before, using HIDR for dimensionality reduction results in the low-level features, that do not carry any semantic meaning. We have two different ways to resolve this issue. The first way is to leverage the encoder gradient method as a *feature selection* tool, as discussed in Section 5. This method selects a subset of the original feature space, where for each cluster of features, only the one that contributes the most to the cluster representation is selected. Hence, this method achieves the dimensionality reduction while keeping the semantic meanings of the original features at the same time. Then we can run either Macrobase or EXstream as an ED method on the selected feature set. We list all the results in Table 7. As we can see, compared to the ED results on the raw input data, Macrobase and EXstream achieve a significant improvement. The quality of their results is indeed comparable to the one using low-level features in Table 6.

EXPT4: Generating Explanations with the Two-Step Approach. As mentioned before, using the HIDR feature selection tool is the first way to run an existing ED method to return interpretable explanations. The second way is to use the two-step HILED approach discussed in Section 6.2. More precisely, we use the aforementioned reduced space to run Macrobase or EXstream to select a set of clusters. Then, on the subset of original features corresponding to these clusters, we run the high-level experiment to generate the explanations for the human user. Running the low-level explanation first provides two main advantages. First, it can reduce the size of the feature space, and take care of the correlation issue among features. Second, it leverages the gradient information which can be used independently in the second step for generating the high-level explanation, or as an optional step between the two steps to further reduce the number of features.

In Table 8, we list the results of the two-step approach with different variants of using the ED method and HIDR feature selection. Since Macrobase achieved a better result in the low-level explanation experiment, in this experiment we always run Macrobase as the first step. The results listed here are for three different combinations. It is worth mentioning that we could not generate the results for Macrobase+Macrobase, since the high-level experiment with Macrobase was still too slow to run. Therefore, we added Gradient as an additional feature selection over the original features. Here, our main observation is that this two-step approach offers high flexibility for the user to try different combinations of an ED method and the Gradient method for feature selection. Also, as we can see in some cases, if properly tuned, the ED methods can lead to results of high quality. For example, compared to Table 7, Marcobase+Gradient and Macrobase+Gradient+Macrobase both lead to reduced values for most of the conciseness and consistency metrics.

Summary. Our main results are as follows: (1) Our HIDR technique outperforms other dimensionality reduction methods in the size of the reduced feature space, as well as in the accuracy of anomaly detection results based on the reduced feature space; (2) None of the three state-of-the-art ED methods can handle high-dimensional feature spaces. Using our HIDR Feature Selection method, these methods can generate explanations with dramatic improvements in conciseness and consistency of the explanations, as well as in efficiency, compared to running ED directly on the raw feature space. (3) Further, our HILED approach provides a flexible framework for trying different ways to combine an ED method with our gradient-based feature selection method, offering an opportunity for the user to fine-tune the performance to achieve better explanations.

8 CONCLUSIONS

In this paper, we presented EXAD for integrated anomaly detection and explanation discovery that allows us to leverage state-of-the-art Deep Learning techniques for anomaly detection, while being able to recover human-readable explanations for detected anomalies. Our key techniques include HIDR dimensionality reduction

	EXstream (low-level)							MacroBase (low-level)							LIME (low-level)						
	Concise		Consistency		Norm.Consis		Time (sec)	Concise		Consistency		Norm.Consis		Time (sec)	Concise		Consistency		Norm.Consis		Time (sec)
	ED1	ED2	ED1	ED2	ED1	ED2	ED1/2	ED1	ED2	ED1	ED2	ED1	ED2	ED1/2	ED1	ED2	ED1	ED2	ED1	ED2	ED1/2
T1	4.00	4.03	1.99	3.50	1.31	2.81	0.0168	1.50	1.38	0.62	2.60	1.17	4.40	0.43	2.60	4.10	2.14	3.27	1.87	2.36	194
T2	1.80	1.86	0.67	2.72	1.11	3.55	0.0065	1.71	1.43	1.04	1.90	1.30	2.60	0.83	3.37	6.57	2.80	3.39	2.13	1.59	179
T3	7.38	8.08	3.03	3.54	1.19	1.43	0.0185	1.70	1.67	0.90	2.32	1.19	3.00	0.31	3.67	6.17	3.01	3.42	2.25	1.74	202
T4	3.73	4.60	1.47	3.63	1.15	2.69	0.0088	1.54	1.55	0.59	3.04	1.06	5.30	0.13	3.44	4.45	2.80	3.22	2.08	2.09	185
T5	2.80	2.43	1.90	3.10	1.59	3.54	0.0056	2.74	2.14	2.17	2.46	1.75	2.57	0.16	3.60	2.00	2.65	2.29	1.77	2.45	167
T6	3.50	3.38	1.73	3.61	1.46	3.61	0.0075	2.17	2.00	1.34	3.16	1.30	4.46	0.11	3.34	3.29	2.60	2.99	2.06	2.43	167
Ave	3.87	4.06	1.80	3.35	1.30	2.94	0.0106	1.90	1.69	1.11	2.58	1.30	3.72	0.33	3.34	4.43	2.67	3.10	2.03	2.11	183

Table 6: Results of low-level explanations returned by EXstream, MacroBase, and LIME when running on the HIDR reduced space of 13 features

	Feature Selection + EXstream							Feature Selection + MacroBase						
	Concise		Consistency		Norm.Consis		Time (sec)	Concise		Consistency		Norm.Consis		Time (sec)
	ED1	ED2	ED1	ED2	ED1	ED2	ED1/2	ED1	ED2	ED1	ED2	ED1	ED2	ED1/2
T1	2.24	2.55	1.02	3.20	1.34	3.61	0.33	1.90	1.86	0.97	2.43	1.16	2.89	0.86
T2	1.57	1.57	0.74	2.30	1.21	3.13	0.50	1.51	1.43	0.55	1.49	1.05	1.96	0.97
T3	3.85	4.75	2.19	3.39	1.56	2.21	0.24	2.93	2.67	1.71	2.03	1.18	1.54	1.0
T4	4.02	3.85	1.48	3.58	1.12	3.10	0.22	1.86	1.85	0.86	2.81	1.13	3.80	0.35
T5	1.77	1.43	1.50	2.65	1.89	4.38	0.03	4.09	3.71	2.55	3.23	1.52	2.52	0.13
T6	3.15	2.75	1.33	3.45	1.07	3.97	0.11	2.50	2.38	1.54	2.75	1.20	2.84	0.44
Ave	2.77	2.82	1.38	3.09	1.37	3.40	0.24	2.46	2.32	1.36	2.46	1.21	2.59	0.63

Table 7: Results of feature selection followed by the ED method, EXstream or MacroBase

	MacroBase+EXstream							MacroBase+Gradient							MacroBase+Gradient+MacroBase						
	Concise		Consistency		Norm.Consis		Time (sec)	Concise		Consistency		Norm.Consis		Time (sec)	Concise		Consistency		Norm.Consis		Time (sec)
	ED1	ED2	ED1	ED2	ED1	ED2	ED1/2	ED1	ED2	ED1	ED2	ED1	ED2	ED1/2	ED1	ED2	ED1	ED2	ED1	ED2	ED1/2
T1	2.05	2.17	1.27	4.99	1.48	14.60	0.69	1.38	1.38	0.33	2.60	1.00	4.40	0.59	1.14	1.14	0.19	2.35	1.07	4.48	0.85
T2	2.26	2.29	1.32	3.58	1.25	5.22	1.12	1.43	1.43	0.43	1.90	1.00	2.60	1.06	1.00	1.00	0.00	1.45	1.00	2.73	1.52
T3	2.38	2.25	1.57	3.93	1.47	6.78	0.50	1.67	1.67	0.63	2.32	1.00	3.00	0.46	1.22	1.08	0.39	1.88	1.21	3.40	0.68
T4	1.82	2.05	1.48	5.02	1.93	15.78	0.28	1.55	1.55	0.51	3.04	1.00	5.30	0.22	1.21	1.15	0.25	2.79	1.04	6.00	0.36
T5	2.77	2.57	1.81	4.17	1.69	7.00	0.27	2.14	2.14	1.02	2.46	1.00	2.57	0.24	1.57	1.71	0.97	2.29	1.65	2.86	0.33
T6	2.88	3.00	2.11	4.42	1.60	7.13	0.19	2.00	2.00	0.84	3.16	1.00	4.46	0.15	1.38	1.38	0.32	2.66	1.00	4.61	0.24
Ave	2.36	2.39	1.59	4.35	1.57	9.42	0.51	1.69	1.69	0.63	2.58	1.00	3.72	0.45	1.25	1.24	0.35	2.24	1.16	4.01	0.66

Table 8: Results of the two-step approach with different variants of using MacroBase or EXstream as the ED method and HIDR feature selection

and feature selection, as well as the HILED approach to explanation discovery. Our main results from the Exathlon [17] benchmark are: (1) HIDR outperforms other dimensionality reduction methods in the size of reduced feature space, as well as in the accuracy of anomaly detection (AD) results based on the reduced feature space. (2) None of the three state-of-the-art ED methods [3, 23, 41] can handle high-dimensional time series data. Using our HIDR feature selection method, these methods can generate explanations with dramatic improvements in conciseness and consistency of the explanations, as well as in computational efficiency. (3) Further, our HILED approach provides a flexible framework for combining existing ED methods with our gradient-based feature selection method, offering an opportunity for the user to fine-tune the performance to achieve better explanations.

Acknowledgments. This work was supported by the European Research Council (ERC) Horizon 2020 research and innovation programme (grant n725561).

REFERENCES

[1] Scipy hierarchical clustering. https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html
[2] J. An and S. Cho. Variational autoencoder based anomaly detection using reconstruction probability. *TR, SNU Data Mining Center: Seoul, Korea*, 2015.
[3] P. Bailis, E. Gan, et al. MacroBase: Prioritizing Attention in Fast Data. In *SIGMOD*, pages 541–556, 2017.
[4] L. E. Bertossi, J. Li, et al. Causality-based explanation of classification outcomes. In *DEEM*, pages 6:1–6:10, 2020.
[5] M. H. Bhuyan, et al. Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys and Tutorials*, 16(1):303–336, 2014.
[6] L. Bontemps, V. L. Cao, et al. Collective anomaly detection based on long short-term memory recurrent neural networks. In *FDSE*, 10018, 141–152, 2016.
[7] L. Cao, J. Wang, et al. Sharing-aware outlier analytics over high-volume data streams. In *SIGMOD*, pages 527–540, 2016.
[8] R. Chalapathy and S. Chawla. Deep Learning for Anomaly Detection: A Survey. *CoRR*, abs/1901.03407, 2019.
[9] V. Chandola, A. Banerjee, et al. Anomaly Detection: A Survey. *ACM Computing Surveys*, 41(3):15:1–15:58, 2009.
[10] Z. Chothia, J. Liagouris, et al. Explaining outputs in modern data analytics. *PVLDB*, 9(4), 2015.
[11] B. Efron, et al. Least angle regression. *The Annals of Statistics*, 32, 2004.
[12] K. El Gebaly, P. Agrawal, et al. Interpretable and informative explanations of outcomes. *PVLDB*, 8(1):61–72, 2014.
[13] N. Frosst and G. E. Hinton. Distilling a neural network into a soft decision tree. In *Int'l Workshop on Comprehensibility and Explanation in AI and ML*, 2017.
[14] M. Gupta, J. Gao, et al. Outlier detection for temporal data: A survey. *TKDE*, 26(9):2250–2267, 2014.
[15] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.
[16] H. Huang and S. P. Kasiviswanathan. Streaming anomaly detection using randomized matrix sketching. *PVLDB*, 9(3):192–203, 2015.
[17] V. Jacob, F. Song, et al. Exathlon: A benchmark for explainable anomaly detection over time series, 2021. https://arxiv.org/abs/2010.05073.
[18] V. Jeyakumar, O. Madani, et al. ExplainIt! - A Declarative Root-cause Analysis Engine for Time Series Data. In *SIGMOD*, pages 333–348, 2019.
[19] H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *KDD*, pages 1675–1684, 2016.
[20] S. M. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *NIPS*, pages 4765–4774, 2017.
[21] M. Ma, Z. Yin, et al. Diagnosing Root Causes of Intermittent Slow Queries in Large-Scale Cloud Databases. *PVLDB*, 13(8):1176–1189, 2020.
[22] P. Malhotra, L. Vig, et al. Long short term memory networks for anomaly detection in time series. In *ESANN*, pages 89–94, 2015.
[23] M. T. Ribeiro, S. Singh, et al. "why should I trust you?": Explaining the predictions of any classifier. In KDD, pages 1135–1144, 2016.
[24] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In AAAI, 2018.
[25] S. Roy, L. Orr, and D. Suciu. Explaining query answers with explanation-ready databases. *PVLDB*, 9(4):348–359, 2015.
[26] T. Schlegl, P. Seeböck, et al. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *IPMI*, pages 146–157, 2017.

[27] scikit-learn documentation: Factor analysis. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.
[28] scikit-learn documentation: Principal component analysis (pca). http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.FactorAnalysis.html.
[29] K. Simonyan, et al. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
[30] F. Song, Y. Diao, et al. EXAD: A System for Explainable Anomaly Detection on Big Data Traces. In *ICDM Workshops*, pages 1435–1440, 2018.
[31] F. Song, B. Zhou, et al. Anomaly Detection and Explanation Discovery on Event Streams. In *BIRTE*, pages 5:1–5:5, 2018.
[32] How are big companies using apache spark. https://medium.com/@tao_66792/how-are-big-companies-using-apache-spark-413743dbbbae.
[33] M. Sundararajan, A. Taly, et al. Axiomatic attribution for deep networks. AAAI, pages 3319–3328, 2017.
[34] J. B. Tenenbaum, V. De Silva, et al. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.
[35] L. Tran, L. Fan, and C. Shahabi. Distance based outlier detection for data streams. *PVLDB*, 9(4):1089–1100, 2015.
[36] D. Van Aken, A. Pavlo, et al. Automatic database management system tuning through large-scale machine learning. In *SIGMOD*, pages 1009–1024, 2017.
[37] E. Wu and S. Madden. Scorpion: Explaining away outliers in aggregate queries. *PVLDB*, 6(8):553–564, 2013.
[38] M. Wu, M. C. Hughes, et al. Beyond sparsity: Tree regularization of deep models for interpretability. AAAI, 2018.
[39] H. Xu, W. Chen, et al. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *WWW*, pages 187–196, 2018.
[40] J. Yang, S. Rahardja, et al. Outlier detection: How to threshold outlier scores? In *AIIPCC*, pages 37:1–37:6, 2019.
[41] H. Zhang, Y. Diao, et al. Exstream: Explaining anomalies in event stream monitoring. In *EDBT*, pages 156–167, 2017.