CONTEXT-INVARIANT, MULTI-VARIATE TIME SERIES REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern time series corpora, in particular those coming from sensor-based data, exhibit characteristics that have so far not been adequately addressed in the literature on representation learning for time series. In particular, such corpora often allow to distinguish between *exogenous* signals that describe a context which influences a given appliance and *endogenous* signals that describe the internal state of the appliance. We propose a temporal convolution network based embedding that improves on the state-of-the-art by incorporating recent advances in contrastive learning to the time series domain and by adopting a multi-resolution approach. Employing techniques borrowed from domain-adversarial learning, we achieve an invariance of the embeddings with respect to the context provided by the exogenous signal. To show the effectiveness of our approach, we contribute new data sets to the research community and use both new as well as existing data sets to empirically verify that we can separate normal from abnormal internal appliance behaviour independent of the external signals in data sets from IoT and DevOps.

1 INTRODUCTION

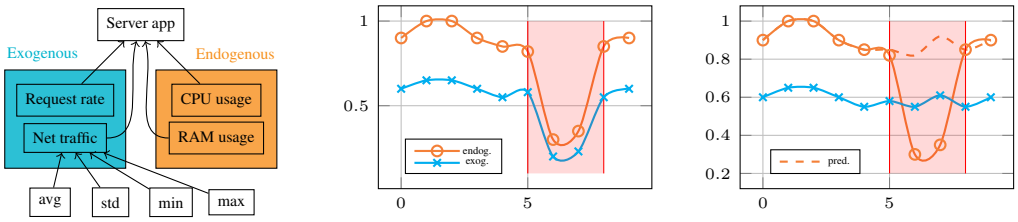
Many modern applications in the physical and virtual world are equipped with sensors that measure the state of the application, its sub-components, and the environment. Examples can be found in the Internet-of-Things (IoT) or in the DevOps/AIOPs space like monitoring wind turbines or cloud-based applications (Lu et al., 2009; Lohrmann & Kao, 2011; Nedelkoski et al., 2019; Li et al., 2020; Krupitzer et al., 2020). Leveraging such time series to identify abnormal appliance behaviour is appealing (see Figures 1b,1c for an overview of time series anomaly types), yet certain characteristics of these time series make them difficult to model with existing representation learning techniques.

First, time series corpora are often highly multi-variate as illustrated in Figure 1a. Each appliance has several sensors[1] associated with it that measure both *exogenous* signals from the environment as well as *endogenous* signals from the internal state of the appliance. Examples for exogenous variables include user behaviour/traffic in a web-based application or physical measurements such as temperature in an IoT context. Conversely, endogenous variables could include the CPU usage or the vibrations of a machine. Increased (application-internal) network traffic is expected with higher user load, and higher ambient temperatures naturally result in elevated temperature of a wind turbine. It is however important to understand when an application deviates from such expected patterns and exhibits unexpected behaviour relative to its environment. We call such effects *contextual anomalies*.

In addition, a defining characteristic of such time series corpora is the sparsity and noisiness of their associated labels. A label could indicate time spans when an application was in an a-typical state. This sparsity may be due to diverse reasons ranging from practical (e.g., data integration or cost of labelling) to fundamental (internal system failures may be exceedingly rare). Noisiness stems from the fact that failures are often subjective and human insight is needed or alarms come from rule-based systems that are themselves overly noisy (Bogatinovski et al., 2021; Wu & Keogh, 2021).

Hence, unsupervised or self-supervised representations of time series are needed that take the characteristics of such modern time series corpora into account. However, while the field of representation learning for sequential data has received considerable attention in domains s.a. natural language processing (NLP) (Lan et al., 2020; Mikolov et al., 2013; Fang et al., 2020; Jaiswal et al., 2021),

[1]Each of these sensors may also measure multiple statistics of the signal (e.g., min, max, avg, std).



(a) **DevOps signal hierarchy example**. Each entity contains exogenous and endogenous sensors and each sensor is comprised of different summary statistics.

(b) **Pattern anomaly**: The time series is anomalous w.r.t its typical behaviour. The depency structure between exogenous and endogenous variables is preserved.

(c) **Contextual anomaly**: The time series is anomalous w.r.t context from another time series. The dependency structure between exogenous and endogenous variables is changed.

Figure 1: Structure of time series corpora (left) and different types of anomaly (middle and right).

similar work in the numerical time series domain remains rare. Specifically, rich, multi-purpose representations facilitating down-stream applications are common in NLP. Instead, feature extraction methods mostly dominate for time series (Lubba et al., 2019; Christ et al., 2017) with Franceschi et al. (2019) providing a notable exception based on temporal convolution networks (TCN). The main contribution of our paper is the extension of this TCN-based approach to cater for the aforementioned complications. We summarize our contributions as follows:

1. We propose *context-invariant* embeddings that allow to identify representations of time series that are invariant to the exogenous variables. We achieve this by adapting domain adversity (Ganin et al., 2016) to the time series domain.
2. We extend the TCN (Franceschi et al., 2019) model with (i) modern contrastive losses that we lift to the time series domain for the first time, (ii) data augmentation techniques, and (iii) considering time series simultaneously at multiple resolutions.[2]
3. We conduct an empirical study in which we show the effectiveness of our approach. We provide a semi-synthetic DevOps data set that we contribute to the research community and consider an under-explored wind turbine dataset (apart from classical synthetic and physical datasets).

Our quantitative results show that context-invariant embeddings indeed represent time series data such that contextual anomalies can be identified in a label-effective way. The qualitative results show that the embeddings allow us to navigate complex data sets in an explorative manner (e.g., considering nearest/farthest neighbours of interesting time series snippets).

2 Representation Learning with Context Invariance

To motivate our approach, consider a simplified system where under normal operation a single endogenous variable y depends instantaneously on a single exogenous signal x via a function $y = g(x) + \varepsilon$, where ε is a noise term. The ideal signal to detect contextual anomalies (those that break this relation) is the residual $\delta := y - g(x)$. Under normal operation this signal carries no information about the exogenous variable and thresholding the magnitude of this residual signal can detect anomalies. Our approach is motivated by this setup but extends it to more complex situations where (i) exogenous & endogenous variables can be multivariate, (ii) the relation stochastic & highly non-linear, and (iii) may depend on the history of the system state. In this case we cannot simply compute a "residual signal", but instead we can try to learn unsupervised representations that are *invariant* to the exogenous variable. This means the embeddings should be independent of the driving signal as long as the endogenous variables respond in a typical manner, which captures some aspect of the residual signal of the toy example. In the following sections we formalize this intuition further and show that context invariance indeed helps detect such anomalies.

Let $Z = \{z_i \in D^T\}_{i=1}^N$ be a set of N equally spaced time series z_i of length at most $T \in \mathbb{N}$ where D is a domain of numerical values. We do not assume time series to be of equal length. We assume

[2]Indeed, time series corpora typically consist of equally-spaced time series (e.g., time series with measurements in 1-min, 5-min or 10-min intervals). This allows us to reason at multiple resolutions.

a decomposition $Z = X \cup Y$ such that time series in X allow to predict time series in Y. We call X the set of environmental/exogenous time series and Y the set of internal/endogenous time series. We assume that it is possible to predict Y from X, but we make no assumption on causality.

The goal of this paper is to map sub-series of Z into a high-dimensional embedding space $\mathbb{R}^M$ which preserves loosely defined properties such as: "normal" time series are close to each other and far away from "abnormal" states. This facilitates down-stream tasks such as time series classification or anomaly detection in a label sparse setting. In particular, our definition of "normal" should be *context-invariant*, that is, only changes in the dependency structure between Y and X should result in large distances in the embedding space. For these tasks, a limited number of labels is available that allows to identify a time span of abnormal behaviour. Typically, the amount of labels is such that a supervised approach is prohibitive and even evaluation may be a challenge.

Our representation learning approach consists of two main components: a *predictor network* g that ties the endogenous and exogenous time series together (either by predicting endogenous from exogenous variables, or vice-versa), and, an *embedding network* f which learns embeddings using contrastive losses. We can combine both in multiple ways. One extreme is a two-step approach where we learn embeddings on the residuals of the predictor network. The other extreme is an end-to-end approach, where we learn embeddings such that the distance between (multivariate) time series is adjusted based on the exogenous variables in a domain-adversarial way. Figure 2 depicts the main components of our approach. For the predictor network, we mainly resort to standard models, so we focus our exposition on the main (novel) components in the following.

2.1 Contrastive, self-supervised, learning of multi-resolution TCN network

The basic building block of our embedding network architecture (Franceschi et al., 2019) consists of stacked temporal dilated causal convolutions (Bai et al., 2018). We have multiple such networks, one per time resolution. We illustrate in Figure 4 the effect of aggregation on the input time series. To obtain a consolidated representation, the concatenated representations are mapped through a neural network. These multi-resolution representations allow the network to encode patterns that are more pronounced in the higher resolutions of the time series in a way that is more effective than an encoder which only operates on a single resolution. We choose resolutions manually as the natural granularities corresponding to the base frequency of the time series we consider in our empirical studies (e.g., seconds, minutes, hours).

Similar to (Franceschi et al., 2019), we rely on a contrastive, self-supervised learning approach to train the embedding network. This crucially relies on a *loss function* and a careful selection of positive $(a, b)_p \in (X, Y)$, reference $(c, d)_c \in (X, Y)$ and negative $(x, y)_n \in (X, Y)$ time series snippets on which to compute the loss terms (depicted in Figure 2). Similar time series should be close to each other and dissimilar time series distant from each other in the embedding space. For an embedding network f_W with parameters W, the loss function takes the following general form:

$$\min_W \text{dist}(f_W((c, d)_c), f_W((a, b)_p)) + \max_W \text{dist}(f_W((x, y)_n, (a, b)_p)) \tag{1}$$

We choose $(a, b)_p, (c, d)_c$ such that $(c, d)_c \supseteq (a, b)_p$ while $(x, y)_n$ is such that $x \cap c \approx \emptyset, y \cap d \approx \emptyset$ (e.g., time snippets at different times and from different elements in the batch). Note further that $(x, y)_n$ is constructed to explicitly break the dependency structure in Z by choosing x to be the exogenous variables at a different time than y. During training, we further augment the examples randomly before feeding them to the TCN network. In particular we apply random jittering, scaling, flipping direction, 2d rotation around a center, permuting random segments, magnitude or time warping (Um et al., 2017) and window slicing or wrapping (Guennec et al., 2016).

Equation (1) is designed to support a variety of contrastive loss functions. Apart from the loss discussed in Franceschi et al. (2019), we rely on other, more recent losses which we describe in the following briefly. These losses, in particular the latter two, aim to avoid collapse of the embeddings while taking practical consideration (e.g., the size of the batch) into account.

The **SimCLR** (Chen et al., 2020) takes two random windows z_A and z_B of a time series and encodes it to get two representations h_A and h_B. It then maximizes the similarity between these two representations from the same time series and dissimilarity between others representations in the batch using the Normalized Temperature-Scaled Cross-Entropy loss (Sohn, 2016) as the distance in (1).

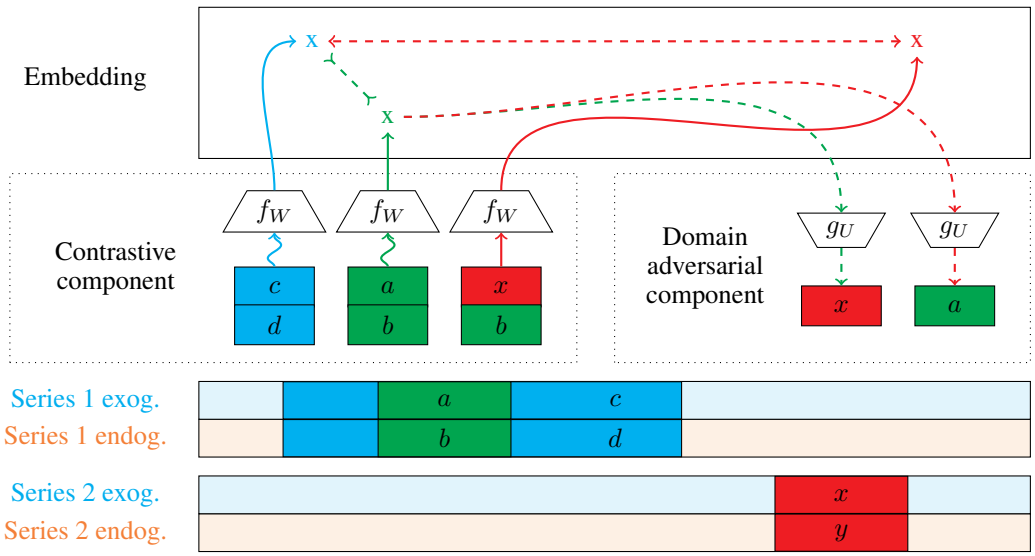


Figure 2: **Context-invariant embeddings with contrastive and domain adversarial learning.** We select negative samples such that the correlation structure between exogenous and endogenous signals is explicitly broken. For the other samples, we augment them randomly, denoted by the squiggly arrows. In addition to the contrastive component, we further add a domain adversarial component that, given a positive embedded sample, tries to reconstruct the context of the positive sample as badly as possible while reconstructing the context of the negative sample as well as possible.

Formally, for temperature parameter τ and mini-batch size of N for each pair we define the loss as:

$$\ell_{A,B} = -\log \frac{\exp(\text{dot}(h_A, h_B)/\tau)}{\sum_{n=1, n\neq A}^{2N} \exp(\text{dot}(h_A, h_n)/\tau)}, \tag{2}$$

where dot is the dot-product between ℓ_2-normalized vectors. This contrastive loss benefits from larger N which might not be feasible in the time series setting and thus we also explore other losses.

Barlow Twins (Zbontar et al., 2021) is a loss operating on two batches of different windows from the same respective time series embeddings, Z_A and Z_B. It computes the cross-correlation matrix along the batch dimension and stores the result in a square matrix C. The final loss then encourages the diagonal terms in this matrix to be close to 1 and the off-diagonal terms be close to 0. Formally,

$$\ell = \sum_i (1 - C_{ii})^2 + \lambda \sum_i \sum_{i \neq j} C_{ij}^2, \tag{3}$$

where $\lambda > 0$ trades off the contribution of the first and second term in the loss. Intuitively, this decorrelation reduces the redundancy between the output embeddings forcing them to contain non-redundant information about the time series.

Unlike SimCLR, **MoCo** (He et al., 2020) uses *two* encoders to obtain representations for the two random windows from the same time series. The representations through the 2nd momentum encoder are preserved in a queue. During training, positive pairs in (2) are constructed from the current batch while negative pairs (denominator of (2) are constructed from the queue of embeddings. The 2nd encoder is updated by linear interpolation of the two encoder with a momentum-based moving average of their weights during training. By using a queue with a slowly changing encoder, this loss attempts to construct large and consistent embeddings which better samples the continuous high dimensional space, independent of the batch size.

2.2 Combinations of Predictor and Embedding Network: from trivial to context-invariant

We can combine the components described above to obtain context-invariant embeddings in a number of ways. The most trivial way is to ignore the predictor network g_U and learn representations of

the entire set Z (or similarly, only for the endogenous part Y). We denote this idea as `BasicEmb`. Next, a two-step approach for context-invariance consists of training the predictor network g_U first, then learn embeddings $g_U(Y) - Y$ on top of the *residual* signals in a second step. We refer to this approach as `ResEmbRegr`. Finally, we describe how to combine both networks into an end-to-end model which we call `ContInvEmb`. This approach is the most flexible in the sense that it allows to adjust the strength of the context-invariance depending on the need of the application.

Our goal is to construct embeddings that are invariant to the corresponding (exogenous) context. Put differently, the embeddings should not contain information that allows prediction of the corresponding exogenous context. This in turn requires to regress the exogenous signals against the embedding but instead of minimizing the regression error, we attempt to maximize the regression error. More formally, we have

$$R(W) = \min_U \sum_{(a,b)_p,(x,y)_n} L_r(W, U, a, b, x, y) + \lambda \max_U \sum_{(a,b)_p} L_r(W, U, a, b)\,, \tag{4}$$

where $L_c(W, U, a, b, x, y)$ is a regression loss, W are the parameters of the encoder network f_W, and U are the free parameters of the prediction network g_U.

The loss in Equation (4) aims to reconstruct the exogenous variables from embeddings as badly as possible, thereby pushing embeddings to invariance wrt the exogenous context. The first loss term leads to good predictions of the exogenous variables shifted with respect to the embedded signals. The second term leads to bad predictions of exogenous variables a from embeddings of positive examples $(a, b)_p$. Gradient reversal handles the adversarial learning aspect (Ganin et al., 2016).

Instead of using to a regression loss in (4), we resort to a multi-class classification problem by discretizing the input space. This has attractive properties in related tasks (Rabanser et al., 2020), but importantly for this particular application avoids explicitly handling trivial predictions (like $g(\cdot) = \pm\infty$) and domains are naturally bounded in the practical applications we consider. Combining (4) with a contrastive loss, we arrive at the following overall loss:

$$\min_W (L(W) + \lambda R(W))\,, \tag{5}$$

where $L(W)$ is a contrastive loss as discussed in Section 2.1, and $R(W)$ is weighted by λ (a hyperparamater) and acts as a regularization term. For $\lambda > 0$, we obtain context-invariant embeddings and for $\lambda = 0$, we recover (Franceschi et al., 2019) (modulo our extensions).

3 Related work

Computer vision (CV) and NLP (Chen et al., 2020; van den Oord et al., 2019; He et al., 2020; Fang et al., 2020; Jaiswal et al., 2021) have embraced self-supervised representations. Most relevant to us, Sohn et al. (2021) learn representations with a contrastive loss to enable anomaly detection in CV. In contrast, time series analysis has not seen a similar adoption of self-supervised techniques for learning general-purpose representations. Franceschi et al. (2019) provide a notable exception by proposing a TCN based embedding (Bai et al., 2018) learnt with a contrastive loss.

The approach of Franceschi et al. (2019) departs from a rich field of feature extraction from time series (Lubba et al., 2019; Christ et al., 2017). While these approaches indeed classify time series well in practice, they mostly focus on the uni-variate case. Their extensions to the multi-variate case are out-performed by Franceschi et al. (2019); Bagnall et al. (2018). Furthermore, the versatility of the features learned by classical approaches is limited by the fact that distances in the induced embeddings are not properly learnt. We extend Franceschi et al. (2019) in the following directions: (i) we adopt it to be multi-resolution; (ii) we equip it with more recent contrastive loss functions and (iii) we turn it into a context-invariant embedding via domain-adversarial learning.

For (i), we note that the de facto choice in a multi-resolution context would be to sub-sample the time series. This is done in classical Wavelet analysis (Mallat, 1989). Instead, we draw inspiration from temporal hierarchical time series analysis (Athanasopoulos et al., 2017) and opt to aggregate time series along the time dimension leading to vectors with a fixed dimension. For (ii), we adopt loss functions (He et al., 2020; Zbontar et al., 2021; Chen et al., 2020) recently proposed for contrastive and self-supervised learning and transfer them to the time series domain. For (iii), we draw

on domain-adversarial representation learning, primarily Ganin et al. (2016). While Ganin et al. (2016) learns embeddings with respect to a specific label classification task, we instead adopt an unsupervised approach via contrastive learning. Further, we replace the domain classifier with an exogenous context regressor whose loss we seek to *maximize*. Such deep prediction networks can be sophisticated, as is the case in particular in the forecasting literature (see e.g., Benidis et al. (2020) for an overview). Our approach readily extends to these, but we restrict ourselves to standard neural regression models as they suffice in the scenarios that we consider for our empirical studies. Other approaches such as hypernetworks (Ha et al., 2016) are conceivable. Yet, they suffer from a lack of computational efficiency and robustness which inhibits their practical applicability.

4 EMPIRICAL EVALUATION

Our empirical evaluation consists of two parts. We first focus on dissecting the improvements on embedding learning through multi-resolution handling, contrastive losses and data-augmentation. Second, we examine the context-invariant representations towards their anomaly detection potential.

4.1 IMPROVEMENTS TO EMBEDDING

For this base experiment, we aim to show the versatility of the embeddings through a down-stream forecasting and classification task each of which we evaluate on two datasets representing the easy and the hard spectrum of the task. We do not aim for comprehensiveness in these first set of experiments but rather for an assessment of the relative improvements through our extensions. As a reference point to gauge the absolute accuracy better, we include a classical feature extraction baseline (Lubba et al., 2019), `Catch22`. In our experiments, we address the downstream classification and forecasting task via simple linear models.

In **classification**, we consider a synthetically generated data set for which we know the labels from the data generation process and the M5 forecasting competition dataset (Theodorou et al., 2021). The latter data set is a retail demand forecasting data set that has product categories associated with the time series. Using a (multinomial) logistic classifier, we aim to predict the labels in both data sets. The synthetic data set is designed such that a high classification accuracy can be and the labels are "objective". In contrast, for the M5 data set, a high classification accuracy cannot be expected. Apart from the amount of product categories available ($\approx$3k), time series associated to different products may have similar characteristics and hence, from a time series classification point of view, labels do not represent a ground truth, a common scenario in practice.

In **forecasting**, we predict electricity[3] and M5 (Makridakis & Spiliotis, 2021) using a shared linear forecaster. The forecaster takes the context embeddings from a large time series window together with a smaller context window of the actual time series to predict for the dataset's target horizon. This horizon is 24 time steps ahead for electricity and 28 days for M5 and compare the metrics with the corresponding test splits.

Discussion. Table 1 summarizes our findings. The more recent proposals for contrastive losses, `MoCo` and `Barlow Twins`, are superior, but there is no clear indication which of both losses is superior overall. While present, we remark that the relative improvements in accuracy are small when compared to the performance wins reported in the original domains for which these losses were designed. We speculate that this may be more attributable to the datasets in the respective domains (and the objectivity of the associated labels) than the loss functions themselves.

Note that including multi-resolution leads to such overwhelming improvements in the classification task that we show only multi-resolution embeddings (for the base resolution, factor 60 and 300) for the classification tasks as these are almost strictly superior ($> 10\%$). Similarly, we report results in the classification task with data augmentation, although improvements are not consistent for data augmentation (Appendix B.2 contains results for an ablation study). The effectiveness of multi-resolution may be surprising as higher capacity models in general and convolutions with a higher dilation in particular should in theory be able to model similar effects. However, adding multiple resolutions offers an inductive bias akin to lagged values in RNN-based forecasting models (e.g., Salinas et al. (2019)) which have been shown to lead to superior practical results.

[3] `https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014`

Loss	Classification ↑: Syn	Classification ↑: M5	Forecasting ↓ : Electricity	Forecasting ↓ : M5
SimCLR	97.00	1.34	0.097	0.70
Barlow Twins	97.25	**5.93**	0.091	0.699
MoCo	**99.50**	3.08	**0.089**	**0.694**
Catch22	64.50	0.03	0.11	0.713
Linear	-	-	0.092	0.698
DeepAR	-	-	0.073	0.90

Table 1: Classification and forecasting accuracy using embeddings learnt with different losses. For forecasting, the accuracy metric shown is P50 loss. Results are averages over 5 runs. The last three lines are baselines using classic time series features, Catch22, and replacing embeddings with more historic values for the forecasting task, Linear and a pure forecasting method, DeepAR.

For forecasting, we note that past historic values instead of embeddings (Linear in Table 1) is a competitive approach, in particular for forecasting the M5 data set. Nevertheless, higher quality embeddings coincide with better forecasting accuracy (comparing Catch22 with our embeddings). Compared with the Electricity dataset, the M5 dataset offers less overall structure so historic values offer a strong signal. Consequently, forecasting accuracy wins are more pronounced in the Electricity dataset.

4.2 Context Invariant Embeddings

In our main experiments, we consider the effect of domain-adversarially learnt context-invariant embeddings in both qualitative and quantitative experiments. The task consists of the identification of contextual anomalies. Perhaps surprisingly, qualitative evaluation is almost more meaningful for the anomaly detection task in general, and for contextual anomalies in particular, given the subjectivity and noisiness of the labels. Note that most publicly available anomaly detection data sets[4] are not suitable for the task, so we do not consider them. Instead, we evaluate on 4 different data sets, each of which allows for a separation into exogenous and endogenous signals. We discuss these data sets first, then the evaluation approach, the models under consideration, and finally discuss the results. Appendix B.3 contains further details.

Datasets The selection of the evaluation datasets aims to balance physical and virtual appliances as well as synthetic and real-world data. Note that synthetic data come with perfect labels, while real-world data typically does not. While lamentable, we believe that this subjectivity and noisiness must be embraced as fundamental in the task. Appendix A contains further details on the data sets.

Synthetic data. We generate a total of 4×360 time series of length 700, based on simple generative models. We generate two exogenous signals, as well as two endogenous signals. We inject two types of anomalies into the data: (i) *pattern anomalies*, i.e., anomalies in the exogenous which are also instantly reflected in the endogenous variables and (ii) *contextual* anomalies only in the endogenous variables. We aim to detect contextual anomalies.

Pendulum. We consider the case of a swinging pendulum with added control signals, where we control the dampening of the acceleration from the outside as an exogenous signal (towards which we want to be invariant) and consider as contextual anomamlies those where we inject an anomalies as a change in the length of the chord which we capture as part of the endogenous signal. Our aim with this data set is to understand how well our models handle cases where the dependency structure between X and Y is more complex.

DevOps. This is a new, semi-synthetic data set[5] that we generated for the purpose of this publication to resemble commonly observed data sets behind corporate firewalls. The object under consideration is a popular cloud-based microservice demo application,[6] which is commonly used in an AIOPs context (Wu et al., 2020). As exogenous signal, we record user interaction approximated by the net-

[4] Wu & Keogh (2021) convincingly argue that many of these datasets should be abandoned.

[5] Jointly with this publication, we open-source both the raw data as well as the set-up to produce the data. To the best of our knowledge, we are the first to extract a data set from this set-up that allows the machine learning community to interact with this area without deep engineering knowledge which is more present in the system's community where the data generation framework is typically considered.

[6] https://github.com/microservices-demo/microservices-demo

work outbound traffic of an application that induces synthetic load on the application and which we fully control. The endogenous signals consists of metrics like CPU, memory and others of the actual microservice application. Each recorded metric has multiple statistics available. We inject anomalies both in the user behavior (leading to pattern anomalies) and in the internal state (contextual anomalies). We want to ignore the former and find the latter. The appendix contains illustrative plots similar to Figure 6. Note that although we have almost total control of the application and its anomalies, the labelling of anomalies is still not perfect, thereby adding further to the complications of public anomaly detection benchmarks (Wu & Keogh, 2021).

Turbine. We consider a wind turbine data set open-sourced[7] by Energias de Portugal. The time series panel can be separated into exogenous and endogenous signals. The former consists of wind speed/direction, ambient temperature and, the pitch angle of the blades (this is controlled from the outside and including it improves the quality of the predictive model). The latter consists of rotation speeds for the turbine and generators, internal temperature on different components, as well as the power output each. All series are available for 4 distinct turbines, are sampled at a 10 minute frequency and are available for 2016. Note that this data set contains only few (43) labelled anomalies and visual inspections of the data reveals inconsistencies with these labels, e.g., some time series appear to be mislabeled (see Sec. 4.2 for an example). The providence of these labels is from automated alarming systems which are often threshold based. Hence, quantitative evaluations cannot be taken at face value. However, given the rich structure of the data and the fuzziness of the task stemming from the labels, the versatility of our approach can be illustrated qualitatively.

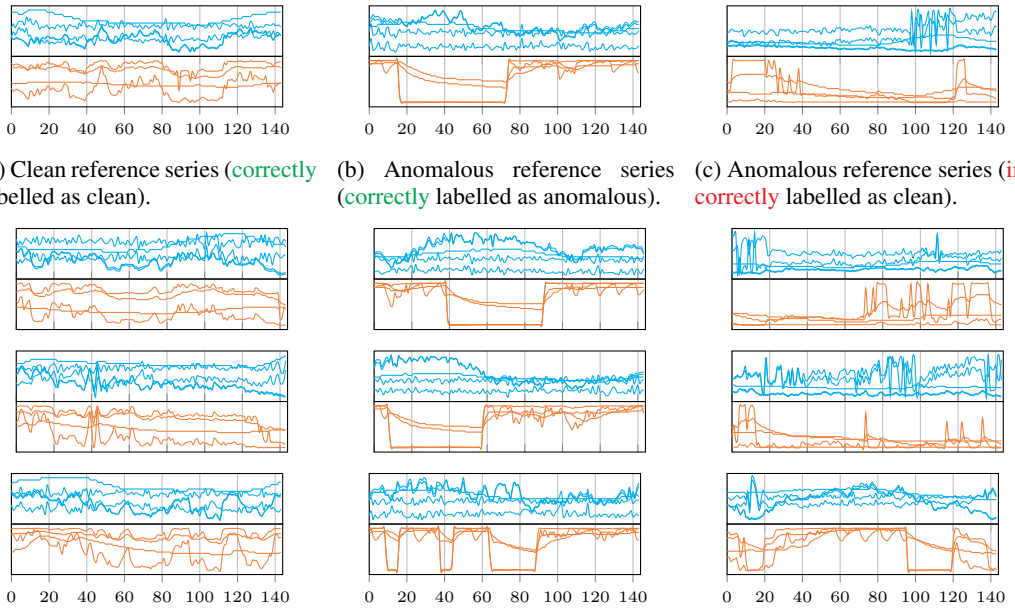
Models & Evaluation We evaluate the following model configurations for their suitability for the contextual anomaly detection task: `BasicEmb`, the modified (Franceschi et al., 2019) (with multi-resolution) learnt ignoring the structure imposed by endogenous and exogenous signals; `ResEmbRegr`, embeddings on the residuals of a predictive models (a feed-forward neural network); and `ContInvEmb`, context-invariant embeddings with a simple linear model for the prediction task (the simplicity ensure that the embeddings are adjusted enough). Moreover, we also provide results for three baseline approaches: `ResTresh`, which computes residuals as in `ResEmbRegr` combined with a simple thresholding mechanism; and two instances of (Lubba et al., 2019): `Catch22` and and `ResCatch22` which compute a feature vector per original and residual time series respecitvely similar to `ResEmbRegr`. The only hyperparameter tuning that we perform is on the Synthetic data set as we do not have enough labels available otherwise.

For all methods, we report standard AUROC scores on the contextual anomaly detection task. For the `ResThresh` method, we compute the AUROC score based on the maximum residual value over the full residual series. For the embedding-based approaches, we use a k-nearest-neighbor classifier in the embedding space to determine a discrete anomaly label.

Quantiative Results Tables 2 summarizes the quantiative results. First, we note that the difficulty of contextual anomaly detection differs widely with the data sets. For example, contextual anomaly detection seems relatively easier on the Turbine data set compared with the DevOps data set despite the latter being semi-synthetically generated with controlled labels. Second, we note that `ContInvEmb` leads to overall superior results when comparing the embedding based approaches (the first three columns in Table 2) and is overall competitive results in many cases, but not always. For the DevOps and Turbine data sets `ResTresh` is overall best by a margin. One explanation is that `ResTresh` best resembles the label generation process by the automated alarming systems on the turbines. For Pendulum data set it is worthwhile to note that `ContInvEmb` delivers an overall superior approach despite the complex non-linear interaction between the exogenous and endogenous signals and only a linear context-predictor component.

Qualitative Results: a case study on wind turbines The embeddings allow to navigate the time series corpus via distances which can be helpful in exploring the data set and uncovering data or label issues. In Figure 3 we show this in qualitative results for context-invariant embeddings. The first two columns show sanity checks: as expected, reference time series (in the top row) labelled as normal or abnormal have as their nearest neighbors (in the same column as the reference time series below it) normal and abnormal time series respectively. Despite the multi-variate nature of the data, visual inspection confirms that the nearest neighbors are plausible. The last column in Figure 3

[7] `https://opendata.edp.com/pages/homepage/`



(a) Clean reference series (correctly labelled as clean).

(b) Anomalous reference series (correctly labelled as anomalous).

(c) Anomalous reference series (incorrectly labelled as clean).

Figure 3: **Nearest-neighbor analysis of context-invariant approach on the Turbine dataset.** Each panel shows exogenous signals on the top and endogenous signals at the bottom. The first row shows (a) a normal reference series; (b) an abnormal reference series (anomaly between time steps 20 and 70); and (c) an anomalous reference series that was mistakenly labeled as clean. Below each of these series we show 3 nearest neighbors in the learned embedding space. For the left and center column, both the correct label and our predicted label match. For series (c), the reference time series is labelled as clean, but its nearest neighbors are abnormal. It is apparent that the erratic spiking patterns in either exogenous/endogenous signals are usually not reflected in the other series.

	`BasicEmb`	`ResEmbRegr`	`ContInvEmb`	`ResTresh`	`Catch22`	`ResCatch22`
Synthetic	0.512 ($\pm$ 0.022)	**1.000** ($\pm$ 0.000)	0.999 ($\pm$ 0.002)	**1.000** ($\pm$ 0.000)	0.494 ($\pm$ 0.008)	**1.000** ($\pm$ 0.000)
Pendulum	0.969 ($\pm$ 0.013)	0.951 ($\pm$ 0.015)	**0.980** ($\pm$ 0.002)	0.510 ($\pm$ 0.000)	0.904 ($\pm$ 0.000)	0.891 ($\pm$ 0.000)
DevOps	0.535 ($\pm$ 0.041)	0.532 ($\pm$ 0.036)	0.587 ($\pm$ 0.007)	**0.619** ($\pm$ 0.000)	0.573 ($\pm$ 0.000)	0.573 ($\pm$ 0.000)
Turbine	0.632 ($\pm$ 0.015)	0.725 ($\pm$ 0.018)	0.736 ($\pm$ 0.022)	**0.845** ($\pm$ 0.000)	0.512 ($\pm$ 0.000)	0.680 ($\pm$ 0.000)

Table 2: AUROC results on the anomaly detection task with 5 seeds. Larger values are better.

shows an example where a reference time series is labelled as normal, but its nearest neighbors consist of time series that are labeled as abnormal. This could point to an issues with the labels. In this case, without further domain knowledge, it may make sense to re-label the reference time series in column (c) as abnormal for the abnormal stretches 0-40 and 100-130.

5 Conclusion

In this work, we presented self-supervised learning of time series embeddings that are invariant with respect to a known and fully-observed context. While the architectures that we presented here lean on techniques invented for computer vision, we make non-trivial contributions to adapt them to the time series domain. For example, we equip our embeddings to consider multiple resolutions of the original sensor signals simultaneously. We observe that the learned embeddings are sensitive to changes of the dependency structure between exogenous and endogenous variables. As confirmed in our evaluation, this allows our approach to learn embeddings that separate dependency-breaking anomalies in the state of the appliance which is the object of interest.

Potential future works could explore techniques from causal discovery (Haufe et al., 2009; Qiu et al., 2020; 2012) to automatically derive an exogenous/endogenous decomposition of the multi-variate time series panel and extensions to causal representation learning (Schölkopf et al., 2021).

6 REPRODUCIBILITY

We use a combination of public datasets and synthetic data sets generated using publicy available code. Method implementations will be open sourced as part of the review process. Finally, we will share the data sets, the models to create the synthetic data and the code with notebooks that allows to reproduce all our results with the final version of this paper.

REFERENCES

George Athanasopoulos, Rob J. Hyndman, Nikolaos Kourentzes, and Fotios Petropoulos. Forecasting with temporal hierarchies. *European Journal of Operational Research*, 262(1):60–74, 2017. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2017.02.046. URL https://www.sciencedirect.com/science/article/pii/S0377221717301911.

Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018, 2018.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018.

Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Bernie Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, Laurent Callot, and Tim Januschowski. Neural forecasting: Introduction and literature overview, 2020.

Jasmin Bogatinovski, Sasho Nedelkoski, Alexander Acker, Florian Schmidt, Thorsten Wittkopp, Soeren Becker, Jorge Cardoso, and Odej Kao. Artificial intelligence for it operations (aiops) workshop white paper, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Maximilian Christ, Andreas W. Kempa-Liehr, and Michael Feindt. Distributed and parallel time series feature extraction for industrial big data applications, 2017.

Hongchao Fang, Sicheng Wang, Meng Zhou, Jiayuan Ding, and Pengtao Xie. Cert: Contrastive self-supervised learning for language understanding, 2020.

Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *arXiv preprint arXiv:1901.10738*, 2019.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. Data augmentation for time series classification using convolutional neural networks, 2016. URL https://halshs.archives-ouvertes.fr/halshs-01357973/.

David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks, 2016.

Stefan Haufe, Guido Nolte, Klaus-Robert Mueller, and Nicole Kraemer. Sparse causal discovery in multivariate time series, 2009.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2020.

Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning, 2021.

Christian Krupitzer, Tim Wagenhals, Marwin Züfle, Veronika Lesch, Dominik Schäfer, Amin Mozaffarin, Janick Edinger, Christian Becker, and Samuel Kounev. A survey on predictive maintenance for industry 4.0. *arXiv preprint arXiv:2002.08224*, 2020.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020.

Yangguang Li, Zhen Ming (Jack) Jiang, Heng Li, Ahmed E. Hassan, Cheng He, Ruirui Huang, Zhengda Zeng, Mian Wang, and Pinan Chen. Predicting node failures in an ultra-large-scale cloud computing platform: An aiops solution. *ACM Trans. Softw. Eng. Methodol.*, 29(2), April 2020. ISSN 1049-331X. doi: 10.1145/3385187. URL https://doi.org/10.1145/3385187.

Björn Lohrmann and Odej Kao. Processing smart meter data streams in the cloud. In *2011 2nd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies*, pp. 1–8, 2011. doi: 10.1109/ISGTEurope.2011.6162747.

Bin Lu, Yaoyu Li, Xin Wu, and Zhongzhou Yang. A review of recent advances in wind turbine condition monitoring and fault diagnosis. In *2009 IEEE Power Electronics and Machines in Wind Applications*, pp. 1–7, 2009. doi: 10.1109/PEMWA.2009.5208325.

Carl H Lubba, Sarab S Sethi, Philip Knaute, Simon R Schultz, Ben D Fulcher, and Nick S Jones. catch22: Canonical time-series characteristics. *Data Mining and Knowledge Discovery*, 33(6): 1821–1852, 2019.

Spyros Makridakis and Evangelos Spiliotis. The m5 competition and the future of human expertise in forecasting. *Foresight: The International Journal of Applied Forecasting*, (60):33–37, 2021.

S.G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989. doi: 10.1109/34.192463.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

Sasho Nedelkoski, Jorge Cardoso, and Odej Kao. Anomaly detection from system tracing data using multimodal deep learning. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pp. 179–186, 2019. doi: 10.1109/CLOUD.2019.00038.

Huida Qiu, Yan Liu, Niranjan A. Subrahmanya, and Weichang Li. Granger causality for time-series anomaly detection. In Mohammed Javeed Zaki, Arno Siebes, Jeffrey Xu Yu, Bart Goethals, Geoffrey I. Webb, and Xindong Wu (eds.), *12th IEEE International Conference on Data Mining, ICDM 2012, Brussels, Belgium, December 10-13, 2012*, pp. 1074–1079. IEEE Computer Society, 2012. doi: 10.1109/ICDM.2012.73. URL https://doi.org/10.1109/ICDM.2012.73.

Juan Qiu, Qingfeng Du, Kanglin Yin, Shuang-Li Zhang, and Chongshu Qian. A causality mining and knowledge graph based method of root cause diagnosis for performance anomaly in cloud applications. *Applied Sciences*, 10(6), 2020. ISSN 2076-3417. doi: 10.3390/app10062166. URL https://www.mdpi.com/2076-3417/10/6/2166.

Stephan Rabanser, Tim Januschowski, Valentin Flunkert, David Salinas, and Jan Gasthaus. The effectiveness of discretization in forecasting: An empirical study on neural time series models, 2020.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2019.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Towards causal representation learning, 2021.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf.

Kihyuk Sohn, Chun-Liang Li, Jinsung Yoon, Minho Jin, and Tomas Pfister. Learning and evaluating representations for deep one-class classification, 2021.

Evangelos Theodorou, Shengjie Wang, Yanfei Kang, Evangelos Spiliotis, Spyros Makridakis, and Vassilios Assimakopoulos. Exploring the representativeness of the m5 competition data, 2021.

Terry T. Um, Franz M. J. Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, Nov 2017. doi: 10.1145/3136755.3136817. URL http://dx.doi.org/10.1145/3136755.3136817.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

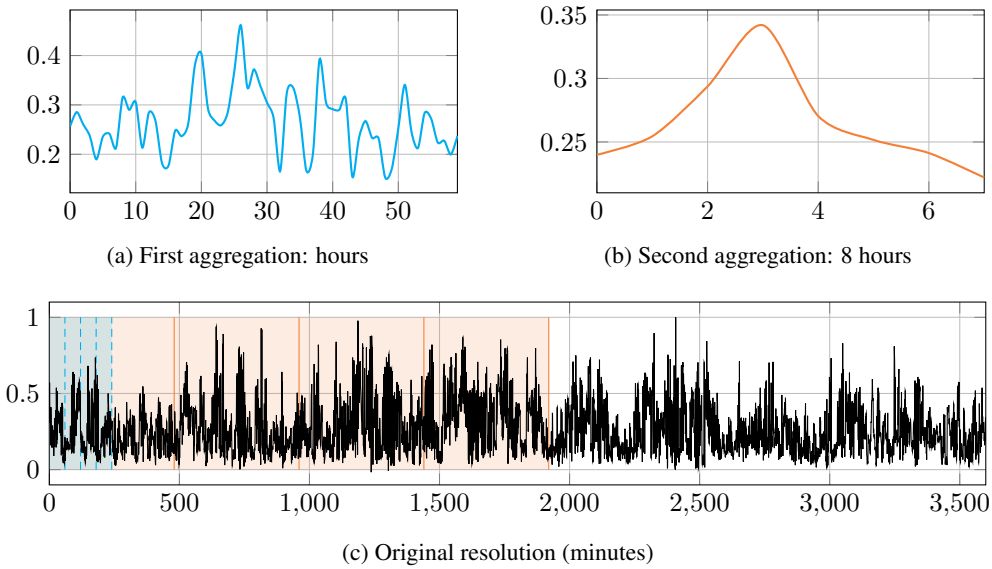
Li Wu, Jasmin Bogatinovski, Sasho Nedelkoski, Johan Tordsson, and Odej Kao. Performance Diagnosis in Cloud Microservices using Deep Learning. In *AIOPS 2020 - International Workshop on Artificial Intelligence for IT Operations*, Dubai, United Arab Emirates, December 2020. URL https://hal.inria.fr/hal-02948735.

Renjie Wu and Eamonn Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021. ISSN 2326-3865. doi: 10.1109/tkde.2021.3112126. URL http://dx.doi.org/10.1109/TKDE.2021.3112126.

Jure Zbontar, Li Jing, Ishan Misra, Yann Lecun, and Stephane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12310–12320. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/zbontar21a.html.

Dataset	Appliances	Entries per appliance	Time steps per entry	Exogenous Variables	Endogenous variables
Synthetic	1 system	360 (1y)	1440 (1min freq) $\rightarrow$ 1d	2	2
Pendulum	1 pendulum	300	144	1	2
Turbine	4 turbines	365 (1y)	144 (10min freq) $\rightarrow$ 1d	5	4
DevOps	1 app	128 ($\approx$11d)	128 (1min freq) $\rightarrow$ $\approx$2h	1	3

Table 3: Data set statistics.



(a) First aggregation: hours

(b) Second aggregation: 8 hours

(c) Original resolution (minutes)

Figure 4: Illustration of the effect of multi-resolution time series aggregations.

A Additional Dataset Details

Table 3 captures details of the datasets under consideration for the contextual anomaly detection task.

Synthetic data We generate two exogenous signals (x_1 and x_2), as well as two endogenous signals (y_1, and y_2) according to the following set of equations.

$$x_1(x) = \sin(\frac{x}{275} - 50) + \sin(\frac{x}{200}) + \epsilon_1 \sim \mathcal{N}(0, 0.05) \tag{6}$$

$$x_2(x) = \sin(\frac{x}{100}) + \epsilon_2 \sim \mathcal{N}(0, 0.1) \tag{7}$$

$$y_1(x) = x_1(x) + \epsilon_3 \sim \mathcal{N}(0, 0.1) \tag{8}$$

$$y_1(x) = x_1(x) + \frac{x_2(x)}{2} - 2 + \epsilon_4 \sim \mathcal{N}(0, 0.08) \tag{9}$$

Turbine While the turbine dataset consists of 82 distinct time series, our experimentation only operates on a subset of these series. In particular, after preprocessing, we select ambient wind speed (`Amb_WindSpeed_Avg`, `Amb_WindSpeed_Est_Avg`), ambient wind direction (`Amb_WindDir_Relative_Avg`, `Amb_WindDir_Abs_Avg`), and ambient temperature (`Amb_Temp_Avg`), as our exogenous variables and generator rotational speed (`Gen_RPM_Avg`), generator power output (`Prod_LatestAvg_TotActPwr`), and two internal temperature sensors (`Gear_Bear_Temp_Avg`, `Hyd_Oil_Temp_Avg`) as our endogenous variables. Our core experiments are based on data from turbine 1.

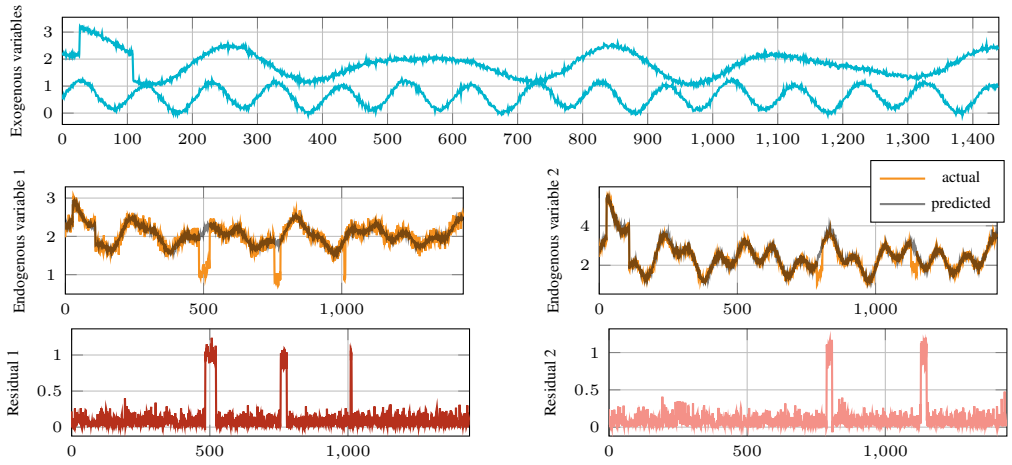


Figure 5: **The obtained residuals based on the Synthetic dataset.** The top panel shows two distinct exogenous variables. For the most part, both series follow a smooth seasonal pattern. The first series shows a noticeable pattern anomaly at the beginning of the series. In the middle panel, we depict two endogenous variables along with their predictions. Since the exogenous variable does give us some information about the pattern anomaly, we can easily predict a similar anomaly for the endogenous variable. However, each endogenous variable exhibits additional anomalies that are deviating from typical behaviour. Considering the residuals in the bottom panel, we can see that the residuals only show these contextual anomalies but disregard the predicable pattern anomaly at the start.

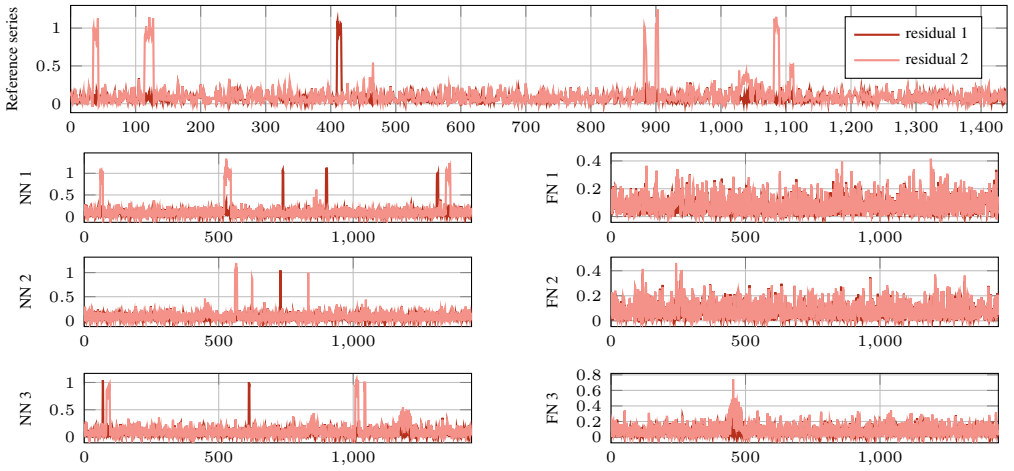


Figure 6: **Nearest-neighbor analysis of `ResEmbRegr` for the Synthetic dataset.** The top panel depicts an anomalous reference series showing the two residual series, both of which exhibit anomalous behaviour as indicated by the residual spikes. By considering the nearest and farthest neighbour series in the embedding space, we can visually inspect the embedding quality. The left panel shows the 3 nearest neighbour series. Clearly, these series align with the reference series since the same spiking pattern in both residuals is present. Considering the farthest neighbours in the right panel, we see that both residual series are noisy without showing pronounced peaks.

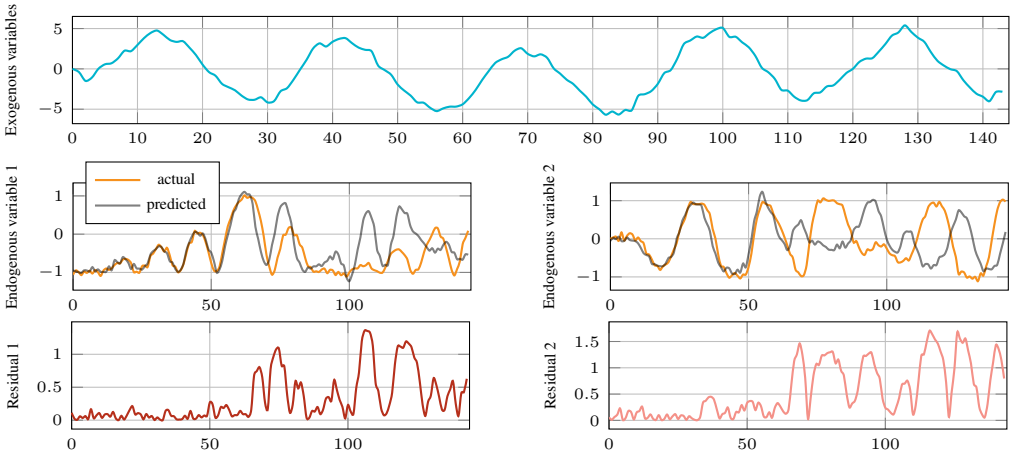


Figure 7: **The obtained residuals based on the Pendulum dataset.** For this data set, we show the control signal in blue (top panel) and see that we cannot predict the endogenous variables well with an instant-effect, multi-variate regression.

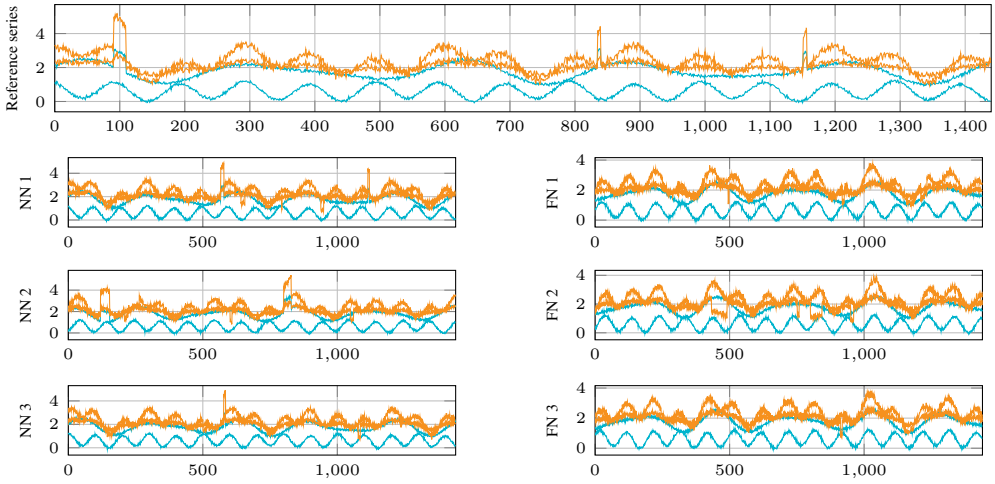


Figure 8: **Nearest-neighbor analysis of `BasicEmb` for the Synthetic dataset .** The top panel depicts a reference series consisting of both the exogenous and the endogenous signals. The endogenous series do not contain any unpredictable anomalies. In contrast to Figure 6, there is no agreement in the nearest neighbors, as the embedding is unable to pick up on the separation into pattern anomalies and contextual anomalies.

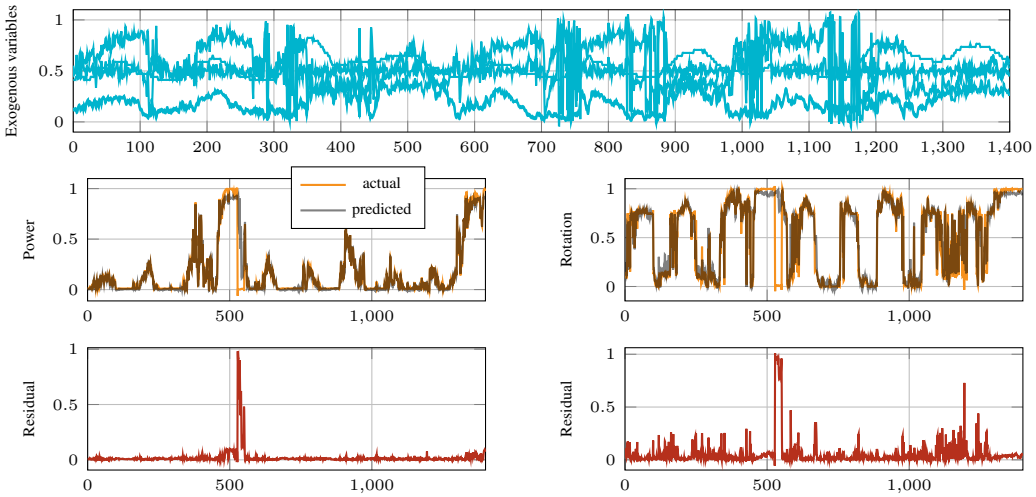


Figure 9: **The obtained residuals based on the Turbine dataset.** Similar to Figure 5.

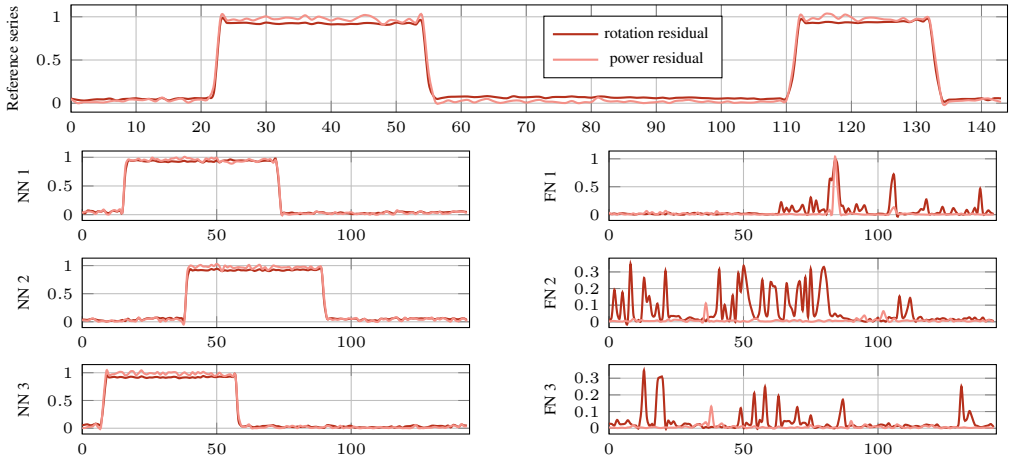


Figure 10: **Nearest-neighbor analysis of `ResEmbRegr` for the Turbine dataset.** Similar to Figure 6

	8/1	16/1	32/1	8/2	16/2	32/2	8/4	16/4	32/4	8/8	16/8	32/8
SR (1)	72.83	74.33	72.50	74.08	78.17	79.83	71.42	72.17	74.25	87.00	89.17	88.42
SR (60)	86.42	86.50	86.08	85.67	**91.75**	92.75	90.83	89.67	87.17	90.50	89.00	83.92
SR (300)	84.33	86.58	88.08	88.25	83.58	86.92	86.58	85.42	83.00	84.58	78.42	82.00
MR MLP (1,60)	80.83	84.83	87.25	82.50	88.08	88.58	86.50	88.92	89.00	87.17	89.42	88.17
MR Stacked (1,60)	90.83	90.83	88.92	90.08	90.08	**94.25**	**93.67**	**91.42**	90.33	92.00	88.58	89.58
MR MLP (1,60,300)	91.67	91.83	91.67	82.33	90.25	92.42	82.17	90.25	**91.75**	89.92	**91.33**	**89.92**
MR Stacked (1,60,300)	**92.17**	**97.08**	**96.58**	**90.83**	91.42	91.75	90.42	90.25	90.17	**92.75**	90.42	88.83

Table 4: Classification accuracy for synthetic data consisting of 4 classes. Rows show single- (**SR**) and multi-resolution (**MR**) models for varying resolutions r. Columns show embedding dimension and number of blocks in the TCN. Provided values are the average accuracy (in %) over 3 random seeds. Bold values indicate best values per column.

B More Experimental Details

B.1 Improvements to Embeddings: forecasting baseline

In order to assess the quality of our embeddings, we use the linear forecasting model depicted in Figure 11.

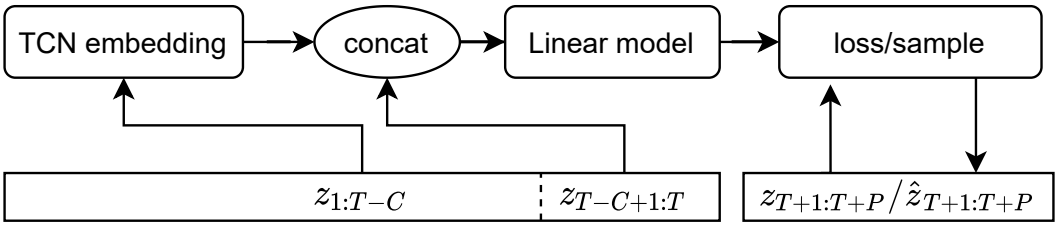


Figure 11: A model to measure the performance of the trained embeddings, where we first embedded a large time series window of size T and then concatenate the resulting embedding with a small C context window sized interval and pass it through a linear model to predict the next prediction window interval P from the training data.

B.2 Improvements to Embeddings: Multi-resolution

Figure 13 provides the overview of the architecture that we consider for multi-resolution embeddings. Considering a time series at different resolutions allows to focus models on different aspects of the time series as Figure 4 depicts. In the following, we evaluate the effect of multiple resolution embeddings (without context-invariance) on synthetic time series data. For sampling a subseries $\boldsymbol{z}^{(1)} \in \mathbb{R}^t$ from a time series $\boldsymbol{z} \in \mathbb{R}^T$ ($t \leq T$) with multiple resolutions, we first sample two indices $i, j \in \{1, \ldots, T\}$ with $i < j$ and $j - i \leq L$ at the "base resolution" $r = 1$. For any resolution $r > 1$, we aggregate $\boldsymbol{z}$ into $\boldsymbol{z}^{[r]} \in \mathbb{R}^{\lceil \frac{T}{r} \rceil}$ by averaging the values from non-overlapping windows of size r. Indices i and j are then converted to the higher resolution r and $\boldsymbol{z}^{(r)}$ is obtained from $\boldsymbol{z}^{[r]}$.

In order to showcase the benefit of leveraging multiple resolutions at the same time, we conduct the following experiment. We generate synthetic data from four different classes of time series. Figure 12 shows these classes which can be distinguished much more easily at resolution $r = 300$ than at $r = 1$. For each class, we generate 100 time series and z-normalize the generated time series. To evaluate the usefulness of embeddings for these time series, we perform 5-fold cross validation and train a multinomial logistic regressor on the embeddings to predict the class of the time series data. Table 4 shows the average classification accuracy when using the embeddings of single- and multi-resolution approaches which clearly shows the benefit of using multiple resolutions.

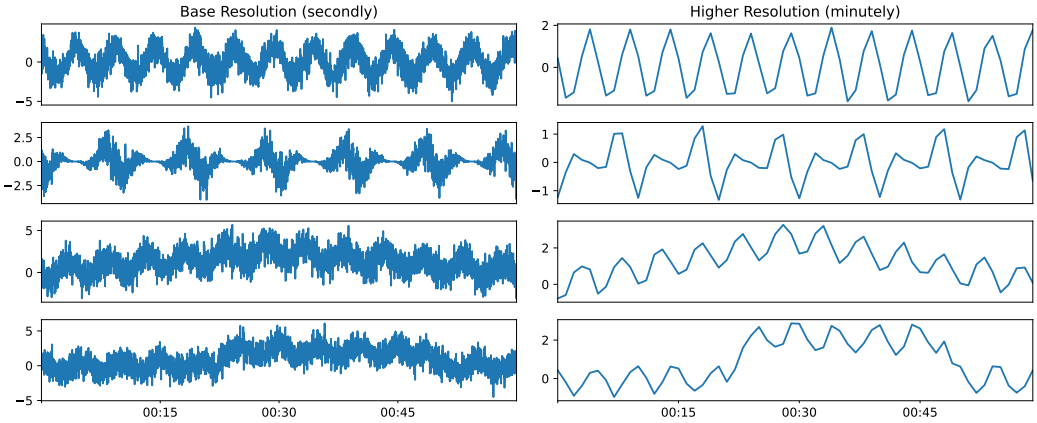


Figure 12: The four different classes of synthetic time series with their base resolution in the left column and a higher resolution on the right column. The topmost class is simple sinusoidal data, the other classes perform the following variations (from top to bottom): (1) amplitude modulation, (2) linear increase and decrease of the mean over the entire time series, (3) shifted mean for a random subseries. Amplitudes, magnitude of shifts and beginning of phase are randomly sampled for each time series.

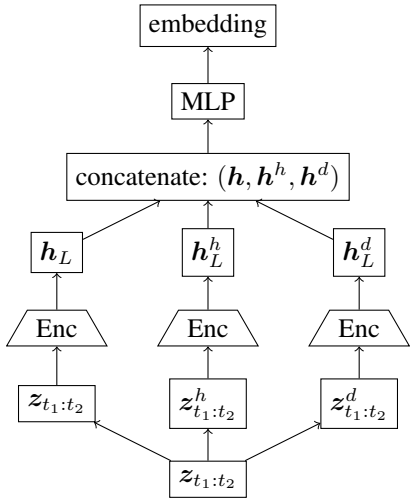


Figure 13: Multi-resolution time series embedding. For the original time series z, different aggregations are computed (e.g., by summation or subsampling), here denoted by z^h (h for hourly) and z^d (d for daily). While we depict three aggregations here, more aggregations are possible. They are then concatenated and combined by an MLP to a final, joint embedding.

	8/1	8/2	8/4	8/8	16/1	16/2	16/4	16/8	32/1	32/2	32/4	32/8
SR (1)	70.50	74.50	76.00	89.83	73.83	79.58	79.58	91.83	77.17	82.25	81.92	94.00
SR (60)	76.33	79.42	80.58	79.75	78.75	82.33	85.25	80.83	79.67	84.58	81.83	84.08
SR (300)	81.42	74.25	80.42	78.92	82.25	84.92	82.75	85.50	87.08	87.58	86.42	88.00
MR Stacked (1,60)	84.08	86.92	90.42	90.08	85.67	89.25	91.42	93.00	87.17	92.50	92.92	93.67
MR MLP (1,60)	61.08	68.83	73.58	82.25	83.17	82.92	86.33	85.33	85.17	87.83	92.50	87.83
MR Stacked (1,60,300)	**90.92**	**90.42**	**94.17**	**93.25**	**93.08**	**97.25**	**96.92**	**96.75**	**97.67**	**97.67**	**98.08**	**98.25**
MR MLP (1, 60, 300)	71.42	73.50	73.17	76.58	89.17	89.75	87.00	91.92	91.92	95.83	92.92	91.42

Table 5: Synthetic Data Accuracy with Barlow Twins and no Data Augmentations.

	8/1	8/2	8/4	8/8	16/1	16/2	16/4	16/8	32/1	32/2	32/4	32/8
SR (1)	71.08	75.00	74.25	89.33	75.17	81.25	80.92	92.67	76.58	82.08	81.00	95.58
SR (60)	73.83	80.67	84.75	77.75	82.33	85.67	88.25	87.33	81.67	85.08	88.25	88.83
SR (300)	73.83	79.42	77.75	80.38	83.83	72.92	79.92	81.00	86.75	86.33	82.92	86.25
MR Stacked (1,60)	87.00	87.00	88.17	90.17	87.58	92.33	91.75	93.92	89.42	93.08	94.83	95.08
MR MLP (1,60)	71.92	62.92	70.58	70.50	78.67	77.08	82.75	80.42	82.33	82.25	86.92	88.67
MR Stacked (1,60,300)	**93.33**	**94.33**	**92.50**	**94.58**	**95.17**	**95.33**	**96.92**	**95.75**	**95.08**	95.75	**96.00**	**95.83**
MR MLP (1, 60, 300)	72.75	65.58	65.75	80.75	86.58	86.75	90.42	82.42	91.67	**97.25**	93.08	93.33

Table 6: Synthetic Data Accuracy with Barlow Twins and Data Augmentations.

B.3 Context Invariant Embeddings: More Results

B.3.1 Hyperparameters

We train all embedding-based methods outlined in Table 2 using batches of size 16 and a learning rate of $1.9 \cdot 10^{-3}$ for 50 epochs. The trained TCN encoder model uses 32 channels, a kernel size of 3 and is comprised of 10 causal convolution blocks. The embedding size is dynamically determined based on the length of the time series (l) and the number exogenous and endogenous dimensions (d_{exog} and d_{endog}) as follows: $l \cdot d_{\text{exog}} \cdot d_{\text{endog}} \cdot 0.1$. The negative and positive selection windows mapping into the embedding space are chosen at random locations and are $l \cdot 0.2$ long. Residuals for all residual-based approaches are obtained using the `MLPRegressor` class from sklearn with out-of-the box parameters. Catch22 is also run using default parameters.

During evaluation, we use $k = 5$ many nearest neighbors to inspect the neighborhood structure induced by the TCN encoder.

Tables 5–22 provide results of a hyperparameter sweep to assess the effect of data augmentations, the different losses and multi-resolution handling.

B.3.2 Qualitative Results

Figure 5 depicts the effect of isolating contextual anomalies via a predictive network on Syntheticdata. In Figures 6, we show the effect of learning embeddings on the residuals vs on the actual values (Figure 8 which shows qualitatively that we need a predictive component to be able to isolate contextual anomalies well). Similar results are available also for the Turbinedataset in Figures 9 and 10. Figure 7 show that purely residual-based approaches can fail in more complex prediction scenarios.

B.3.3 Quantiative Results

For all methods that produce a mapping onto an embedding space, we can further determine whether any particular (embedded) time series is closer to the mean embedding of the respective correct/incorrect class. Concretely, we can compute the distance gap between the distance to the correct class for the respective series and the distance to the incorrect class and average this quantity over all series in the test set. Embedding space gap results are presented in Table 23. This complements the AUROC results in Table 2 by providing an impression of the embedding space and the induced nearest neighborhood structure. More fine-grained results are shown in Table 24.

	8/1	8/2	8/4	8/8	16/1	16/2	16/4	16/8	32/1	32/2	32/4	32/8
SR (1)	72.92	74.92	69.50	86.92	74.25	79.33	73.92	88.67	73.00	79.58	73.42	88.12
SR (60)	85.42	85.75	91.75	91.08	86.83	**91.42**	89.92	86.50	85.83	92.25	86.83	84.50
SR (300)	83.75	88.33	86.58	83.83	86.33	83.83	84.08	80.83	88.17	86.92	85.17	81.92
MR Stacked (1,60)	90.83	90.92	**93.42**	**92.58**	90.92	90.42	**92.25**	89.42	89.17	**94.67**	91.50	88.92
MR MLP (1,60)	80.75	82.67	86.92	88.08	84.42	88.33	88.58	89.67	86.83	88.42	89.08	88.00
MR Stacked (1,60,300)	**92.08**	**93.75**	89.92	92.50	**96.33**	91.33	90.75	90.42	**96.67**	91.75	**91.67**	89.08
MR MLP (1, 60, 300)	91.33	83.75	79.58	89.42	92.25	90.25	89.92	**91.00**	92.33	92.08	91.25	**90.17**

Table 7: Synthetic Data Accuracy with NXTent Loss and no Data Augmentations.

	8/1	8/2	8/4	8/8	16/1	16/2	16/4	16/8	32/1	32/2	32/4	32/8
SR (1)	75.00	78.58	83.83	90.50	74.63	78.25	78.92	**92.50**	76.50	81.08	80.58	91.08
SR (60)	91.42	86.33	92.83	92.00	91.08	84.58	90.58	87.50	90.08	**93.83**	88.83	85.58
SR (300)	86.08	87.25	84.33	81.75	86.67	86.17	85.17	82.42	86.42	86.25	80.75	84.83
MR Stacked (1,60)	90.83	90.67	**94.42**	93.50	89.50	93.42	**92.00**	90.50	91.58	92.00	86.92	89.33
MR MLP (1,60)	88.58	79.75	87.58	88.25	82.25	86.17	85.00	90.08	86.38	86.67	88.75	88.33
MR Stacked (1,60,300)	**94.08**	**92.92**	90.00	**93.87**	**94.25**	**95.50**	89.08	91.42	**97.00**	90.75	**91.83**	88.75
MR MLP (1, 60, 300)	92.33	78.33	77.00	82.33	90.08	89.67	91.92	92.42	94.67	92.75	90.67	**92.08**

Table 8: Synthetic Data Accuracy with NXTent and Data Augmentations.

	8/1	8/2	8/4	8/8	16/1	16/2	16/4	16/8	32/1	32/2	32/4	32/8
SR (1)	75.75	72.00	79.50	88.42	76.33	77.33	79.08	90.67	78.62	76.33	78.83	93.00
SR (60)	92.25	91.17	88.00	92.00	93.58	96.42	94.75	93.75	94.50	97.25	95.92	94.25
SR (300)	88.75	88.00	86.50	86.33	90.33	88.75	87.83	87.75	89.58	89.92	88.83	89.25
MR Stacked (1,60)	93.83	94.33	96.75	92.42	94.67	94.08	94.58	95.75	94.00	97.58	96.17	96.75
MR MLP (1,60)	86.83	80.83	86.58	80.08	88.50	91.58	91.08	90.25	94.58	92.17	93.58	89.92
MR Stacked (1,60,300)	**97.58**	**98.75**	**98.25**	**96.42**	**98.25**	**97.92**	**97.00**	**97.42**	**98.92**	**98.75**	**97.50**	**97.83**
MR MLP (1, 60, 300)	91.08	88.83	91.83	86.50	94.92	94.00	90.13	94.50	96.83	94.75	96.42	96.67

Table 9: Synthetic Data Accuracy with MOCO and no Data Augmentations.

	8/1	8/2	8/4	8/8	16/1	16/2	16/4	16/8	32/1	32/2	32/4	32/8
SR (1)	76.00	75.75	78.50	85.67	76.83	77.83	78.42	91.58	76.08	77.00	80.25	92.25
SR (60)	91.75	91.08	87.17	89.75	93.50	95.75	93.25	93.08	94.50	96.42	95.75	94.83
SR (300)	88.17	86.17	86.58	84.92	89.42	87.83	88.25	89.75	89.83	88.67	89.67	89.25
MR Stacked (1,60)	93.75	93.38	96.08	97.00	94.25	93.75	96.83	94.75	93.75	97.00	**97.42**	96.00
MR MLP (1,60)	87.33	83.42	87.25	83.75	89.17	91.75	91.33	89.58	92.67	92.67	92.75	90.00
MR Stacked (1,60,300)	**97.92**	**98.92**	**98.25**	**97.25**	**98.33**	**98.83**	**97.33**	**97.50**	**98.58**	**99.50**	96.75	**98.17**
MR MLP (1, 60, 300)	90.67	88.92	89.00	87.92	93.83	95.00	93.92	93.75	97.92	94.67	97.00	94.92

Table 10: Synthetic Data Accuracy with MOCO and Data Augmentations.

	32/1	32/2	32/4	32/8	64/1	64/2	64/4	64/8	128/1	128/2	128/4	128/8
SR (1)	0.21	0.18	0.20	0.53	0.19	0.27	0.49	1.50	0.32	0.52	0.99	3.85
SR (60)	0.96	0.72	1.01	0.86	1.34	**1.31**	1.63	1.46	**2.09**	2.37	2.95	2.43
SR (300)	1.16	0.58	0.53	0.49	**1.59**	1.09	1.16	0.92	2.05	1.82	2.01	1.65
MR Stacked (1,60)	0.62	0.41	0.76	0.94	0.79	0.72	1.40	2.29	0.91	1.69	2.56	4.08
MR MLP (1,60)	0.68	0.69	0.76	0.82	0.66	0.76	0.83	1.10	0.76	0.93	0.99	1.13
MR Stacked (1,60,300)	**1.18**	**0.79**	**1.15**	**1.66**	1.20	1.29	**2.24**	**3.11**	1.59	**2.72**	**3.60**	**5.93**
MR MLP (1, 60, 300)	0.71	0.73	0.85	0.85	0.88	0.95	1.08	1.04	1.32	1.17	1.56	1.65

Table 11: M5 Accuracy with Barlow Twins and Data Augmentations.

	32/1	32/2	32/4	32/8	64/1	64/2	64/4	64/8	128/1	128/2	128/4	128/8
SR (1)	0.28	0.34	0.37	1.23	0.38	0.53	0.70	1.84	0.59	0.58	0.92	2.82
SR (60)	1.23	1.35	1.46	1.56	1.72	2.10	2.69	2.25	2.14	3.50	**4.57**	3.65
SR (300)	1.51	0.90	0.76	0.82	1.66	1.40	1.41	1.32	2.30	2.40	2.21	1.87
MR Stacked (1,60)	1.01	0.81	1.28	1.71	1.41	1.59	2.13	2.68	1.85	2.41	3.16	**4.49**
MR MLP (1,60)	0.72	0.87	1.18	1.09	0.98	1.02	1.15	1.51	1.31	1.51	1.64	1.35
MR Stacked (1,60,300)	**2.07**	**1.45**	**1.72**	**2.51**	**2.68**	**3.00**	**3.15**	**3.62**	**3.72**	**4.64**	4.27	4.18
MR MLP (1, 60, 300)	0.87	0.99	1.12	1.30	1.44	1.43	1.69	1.77	2.26	1.72	2.30	2.13

Table 12: M5 Accuracy with Barlow Twins and no Data Augmentations.

	32/1	32/2	32/4	32/8	64/1	64/2	64/4	64/8	128/1	128/2	128/4	128/8
SR (1)	0.34	0.40	0.46	0.91	0.33	0.38	0.56	1.07	0.35	0.42	0.59	1.11
SR (60)	1.40	1.93	2.32	2.48	1.47	1.91	**3.08**	2.45	1.56	2.19	**2.87**	2.56
SR (300)	1.96	1.91	1.86	1.97	2.05	2.13	2.20	1.88	2.09	2.33	2.18	1.93
MR Stacked (1,60)	1.32	1.77	1.96	2.51	1.26	1.46	2.19	2.39	1.28	1.81	1.88	2.30
MR MLP (1,60)	1.15	1.45	1.58	1.58	1.33	1.45	1.78	1.99	1.40	1.57	1.75	1.79
MR Stacked (1,60,300)	**2.77**	**3.04**	**2.70**	**2.99**	**2.75**	**2.71**	2.80	**3.00**	**2.69**	**3.05**	2.76	**2.94**
MR MLP (1, 60, 300)	2.08	1.94	1.93	2.15	2.35	2.25	2.26	2.32	2.29	2.24	2.32	2.33

Table 13: M5 Accuracy with MOCO and Data Augmentations.

	32/1	32/2	32/4	32/8	64/1	64/2	64/4	64/8	128/1	128/2	128/4	128/8
SR (1)	0.41	0.42	0.51	0.79	0.39	0.42	0.52	0.87	0.40	0.46	0.53	0.88
SR (60)	1.83	2.38	**3.02**	**3.05**	1.93	**2.70**	**3.39**	**3.06**	2.20	2.75	**3.65**	**3.38**
SR (300)	2.37	2.36	2.25	2.27	2.51	2.49	2.64	2.36	**2.61**	2.53	2.47	2.48
MR Stacked (1,60)	1.38	1.74	1.81	2.28	1.26	1.57	1.99	2.09	1.25	1.73	1.77	2.01
MR MLP (1,60)	1.27	1.49	1.77	1.67	1.47	1.59	1.86	2.08	1.57	1.73	1.91	1.87
MR Stacked (1,60,300)	**2.61**	**2.97**	2.42	2.65	**2.55**	2.68	2.44	2.72	2.50	**2.85**	2.54	2.60
MR MLP (1, 60, 300)	2.21	2.06	2.03	1.97	2.54	2.46	2.35	2.44	2.60	2.69	2.50	2.38

Table 14: M5 Accuracy with MOCO and no Data Augmentations.

	32/1	32/2	32/4	32/8	64/1	64/2	64/4	64/8	128/1	128/2	128/4	128/8
SR (1)	0.36	0.40	0.56	1.03	0.38	0.44	0.64	**0.98**	0.38	0.44	0.58	1.06
SR (60)	0.74	0.69	0.78	0.71	0.71	0.81	0.85	0.77	0.74	0.78	0.91	0.72
SR (300)	0.82	0.77	0.66	0.60	0.73	0.68	0.62	0.58	0.77	0.75	0.67	0.56
MR Stacked (1,60)	0.82	0.83	0.96	1.06	0.75	0.96	0.81	0.97	0.67	1.02	0.89	**1.15**
MR MLP (1,60)	0.67	0.85	0.80	0.80	0.65	0.83	0.79	0.88	0.70	0.76	0.77	0.84
MR Stacked (1,60,300)	**1.28**	**1.04**	**1.04**	**1.06**	**1.34**	**1.05**	**0.98**	0.94	**1.12**	**1.16**	**0.98**	1.01
MR MLP (1, 60, 300)	0.84	0.79	0.81	0.82	0.85	0.75	0.83	0.79	0.86	0.85	0.79	0.85

Table 15: M5 Accuracy with NTXent and Data Augmentations.

	32/1	32/2	32/4	32/8	64/1	64/2	64/4	64/8	128/1	128/2	128/4	128/8
SR (1)	0.44	0.48	0.63	1.06	0.42	0.54	0.78	1.10	0.47	0.51	0.74	1.09
SR (60)	0.84	0.75	0.92	0.82	0.86	0.89	0.91	0.79	0.84	0.89	0.90	0.90
SR (300)	0.88	0.84	0.80	0.65	0.87	0.83	0.67	0.68	0.84	0.85	0.68	0.67
MR Stacked (1,60)	0.94	1.02	1.12	1.19	0.88	1.12	1.16	**1.25**	0.95	0.99	1.10	1.19
MR MLP (1,60)	0.78	1.05	1.01	0.81	0.88	0.98	0.87	0.91	0.84	0.91	0.94	0.87
MR Stacked (1,60,300)	**1.43**	**1.21**	**1.24**	**1.33**	**1.54**	**1.14**	**1.37**	1.17	**1.10**	**1.19**	**1.25**	**1.25**
MR MLP (1, 60, 300)	1.11	1.10	0.97	0.93	1.08	1.05	0.93	0.98	1.07	1.04	0.94	0.97

Table 16: M5 Accuracy with NTXent and no Data Augmentations.

	32/1	32/2	32/4	32/8	64/1	64/2	64/4	64/8	128/1	128/2	128/4	128/8
SR (1)	2.16	2.65	2.93	7.37	2.90	3.85	4.68	9.83	3.91	4.12	5.48	14.94
SR (60)	6.99	7.51	7.46	8.77	9.13	10.72	12.60	11.39	11.02	15.47	**18.20**	15.46
SR (300)	8.05	5.78	5.45	5.11	9.13	8.10	7.63	6.67	11.36	11.53	10.39	9.40
MR Stacked (1,60)	5.84	5.17	8.18	9.02	7.83	9.10	11.26	12.89	10.18	11.83	15.63	**17.88**
MR MLP (1,60)	4.40	5.18	5.90	6.44	5.44	5.86	6.94	10.04	7.51	8.24	9.52	8.15
MR Stacked (1,60,300)	**10.58**	**8.55**	**8.55**	**12.32**	**13.30**	**14.26**	**13.30**	**16.26**	**15.53**	**17.65**	16.59	16.90
MR MLP (1, 60, 300)	5.85	5.65	6.33	7.33	8.28	8.03	9.56	11.02	10.18	11.08	11.30	10.50

Table 17: M5 Top-10 Accuracy with Barlow Twins and no Data Augmentations.

	32/1	32/2	32/4	32/8	64/1	64/2	64/4	64/8	128/1	128/2	128/4	128/8
SR (1)	1.62	1.22	1.58	3.24	1.40	2.37	3.16	7.67	2.28	3.83	5.63	13.22
SR (60)	5.42	4.84	5.66	4.87	7.70	**7.48**	8.13	7.76	10.30	11.50	14.25	11.86
SR (300)	**7.47**	**4.86**	3.83	4.08	**8.69**	5.71	6.01	5.70	**10.68**	8.98	9.21	7.50
MR Stacked (1,60)	3.73	2.91	5.62	6.04	4.61	5.91	7.84	11.50	5.74	9.36	12.50	16.50
MR MLP (1,60)	4.11	4.46	4.72	4.68	4.14	4.41	5.60	6.13	4.45	5.35	6.18	6.43
MR Stacked (1,60,300)	6.34	4.73	**6.22**	**9.76**	6.89	7.34	**11.23**	**12.83**	8.33	**13.20**	**14.80**	**19.42**
MR MLP (1, 60, 300)	4.42	4.51	5.37	5.52	5.47	5.71	6.16	6.95	6.97	8.18	8.43	8.37

Table 18: M5 Top-10 Accuracy with Barlow Twins and Data Augmentations.

	32/1	32/2	32/4	32/8	64/1	64/2	64/4	64/8	128/1	128/2	128/4	128/8
SR (1)	2.90	3.00	3.48	5.11	2.90	3.20	3.48	5.42	2.97	3.08	3.44	5.63
SR (60)	10.00	12.00	**14.19**	**14.07**	10.54	**13.20**	**14.96**	**14.37**	11.29	13.49	**15.50**	**15.20**
SR (300)	12.01	11.48	11.14	11.32	12.35	12.03	12.43	11.55	12.69	12.17	11.80	11.96
MR Stacked (1,60)	8.04	10.00	9.49	11.96	7.63	8.90	10.46	11.01	7.63	9.66	9.45	10.21
MR MLP (1,60)	8.10	9.08	10.27	9.49	8.84	9.54	10.54	11.42	9.39	10.41	10.45	10.59
MR Stacked (1,60,300)	**12.79**	**13.93**	11.97	13.04	12.73	12.82	12.19	13.28	12.27	13.37	12.46	12.56
MR MLP (1, 60, 300)	11.92	11.75	10.85	11.45	**13.28**	12.47	12.30	12.72	**13.62**	**13.86**	12.69	12.10

Table 19: M5 Top-10 Accuracy with MOCO and no Data Augmentations.

	32/1	32/2	32/4	32/8	64/1	64/2	64/4	64/8	128/1	128/2	128/4	128/8
SR (1)	2.63	3.03	3.37	5.87	2.69	3.23	3.60	6.41	2.70	3.13	3.76	7.04
SR (60)	8.32	9.86	12.34	11.74	8.54	10.15	13.65	11.85	9.01	11.09	**13.20**	12.60
SR (300)	10.27	10.03	9.87	10.24	10.77	10.80	10.96	9.82	10.98	11.34	10.95	9.98
MR Stacked (1,60)	7.92	9.44	10.42	12.78	7.68	8.90	11.25	11.99	7.79	9.60	10.27	11.99
MR MLP (1,60)	7.61	8.87	10.17	9.49	8.60	8.96	10.53	11.55	8.88	10.16	10.27	11.02
MR Stacked (1,60,300)	**13.41**	**14.11**	**13.13**	**13.86**	**13.71**	**13.39**	**13.72**	**14.73**	**13.26**	**14.20**	13.01	**13.75**
MR MLP (1, 60, 300)	11.53	11.31	11.03	11.91	12.86	12.11	11.91	12.82	12.67	12.33	12.16	12.37

Table 20: M5 Top-10 Accuracy with MOCO and Data Augmentations.

	32/1	32/2	32/4	32/8	64/1	64/2	64/4	64/8	128/1	128/2	128/4	128/8
SR (1)	3.21	3.56	4.29	6.32	3.12	3.68	5.06	6.33	3.42	3.81	4.84	6.66
SR (60)	5.24	4.33	5.52	4.94	5.47	5.56	5.60	4.82	5.33	5.47	5.39	5.69
SR (300)	5.25	4.99	4.48	4.17	5.32	4.92	4.35	4.13	4.99	4.69	4.36	4.26
MR Stacked (1,60)	5.83	6.10	6.61	6.64	5.66	6.20	6.44	**7.00**	5.84	6.04	6.38	6.79
MR MLP (1,60)	5.15	6.09	5.94	5.17	5.26	5.85	5.44	5.64	5.21	5.76	5.62	5.64
MR Stacked (1,60,300)	**8.02**	**6.99**	**7.02**	**7.49**	**8.54**	**6.56**	**7.19**	6.83	**6.61**	**6.86**	**7.10**	**6.95**
MR MLP (1, 60, 300)	6.72	6.48	5.84	5.61	6.63	6.29	5.60	5.82	6.22	6.13	5.73	6.02

Table 21: M5 Top-10 Accuracy with NTXent and no Data Augmentations.

	32/1	32/2	32/4	32/8	64/1	64/2	64/4	64/8	128/1	128/2	128/4	128/8
SR (1)	2.85	3.13	3.87	6.21	2.83	3.13	4.13	6.01	2.93	3.12	4.04	6.13
SR (60)	4.71	4.41	4.79	4.31	4.78	5.04	5.39	4.55	4.68	4.92	5.23	4.34
SR (300)	4.88	4.87	4.07	3.85	4.88	4.55	4.05	3.82	4.73	4.52	4.07	3.85
MR Stacked (1,60)	5.05	5.40	5.63	5.83	4.96	5.74	5.53	**6.10**	4.61	5.38	5.36	**6.38**
MR MLP (1,60)	4.38	5.57	5.15	5.05	4.35	5.34	5.01	5.51	4.56	5.12	5.11	5.23
MR Stacked (1,60,300)	**7.61**	**6.26**	**6.25**	**6.25**	**7.68**	**6.25**	**6.60**	5.74	**6.94**	**6.32**	**5.90**	6.08
MR MLP (1, 60, 300)	5.89	5.40	5.31	5.04	5.76	5.26	5.01	5.19	5.61	5.40	5.12	5.35

Table 22: M5 Top-10 Accuracy with NTXent and Data Augmentations.

	`BasicEmb`	`ResEmbRegr`	`ContInvEmb`	`Catch22`
Synthetic	0.051 ($\pm$ 0.041)	0.641 ($\pm$ 0.314)	1.944 ($\pm$ 0.039)	0.008 ($\pm$ 0.010)
Pendulum	0.446 ($\pm$ 0.059)	0.433 ($\pm$ 0.074)	0.186 ($\pm$ 0.178)	0.019 ($\pm$ 0.0)
DevOps	0.014 ($\pm$ 0.006)	0.035 ($\pm$ 0.008)	0.042 ($\pm$ 0.007)	0.013 ($\pm$ 0.0)
Turbine	0.075 ($\pm$ 0.007)	0.319 ($\pm$ 0.103)	0.250 ($\pm$ 0.085)	0.004 ($\pm$ 0.0)

Table 23: The ℓ_2 distance between (normalized) embeddings between correct and incorrect classes on the anomaly detection task with 5 random seeds. Larger values are better.

	BasicEmb			ResEmbRegr			ContInvEmb		
Dataset	Corr	Incorr	Gap	Corr	Incorr	Gap	Corr	Incorr	Gap
Synthetic	0.483 (± 0.249)	0.534 (± 0.257)	0.051 (± 0.041)	0.634 (± 0.216)	1.284 (± 0.122)	0.641 (± 0.314)	0.034 (± 0.012)	1.978 (± 0.029)	1.944 (± 0.039)
DevOps	0.918 (± 0.038)	0.932 (± 0.034)	0.446 (± 0.059)	0.942 (± 0.016)	0.977 (± 0.020)	0.433 (± 0.074)	0.518 (± 0.133)	0.529 (± 0.130)	0.186 (± 0.178)
Pendulum	0.691 (± 0.037)	1.137 (± 0.076)	0.014 (± 0.006)	0.679 (± 0.036)	1.112 (± 0.065)	0.035 (± 0.008)	0.215 (± 0.178)	0.437 (± 0.346)	0.012 (± 0.007)
Turbine	0.927 (± 0.024)	1.002 (± 0.022)	0.075 (± 0.007)	0.691 (± 0.186)	1.010 (± 0.272)	0.319 (± 0.103)	0.265 (± 0.079)	0.516 (± 0.161)	0.250 (± 0.085)

Table 24: Distance results on the anomaly detection task with 5 random seeds. Distances to the correct class (Corr) should be small while distances to the incorrect class (Incorr) should be large. We prefer approaches with a large gap between the mean correct disatance and the mean incorrect distance.