Neural Contextual Anomaly Detection for Time Series

Chris U. Carmona [*] [†]
University of Oxford
carmona@stats.ox.ac.uk

François-Xavier Aubet [*]
AWS AI Labs
aubetf@amazon.com

Valentin Flunkert
AWS AI Labs
flunkert@amazon.com

Jan Gasthaus
AWS AI Labs
gasthaus@amazon.com

Abstract

We introduce Neural Contextual Anomaly Detection (NCAD), a framework for anomaly detection on time series that scales seamlessly from the unsupervised to supervised setting, and is applicable to both univariate and multivariate time series. This is achieved by effectively combining recent developments in representation learning for multivariate time series, with techniques for deep anomaly detection originally developed for computer vision that we tailor to the time series setting. Our window-based approach facilitates learning the boundary between normal and anomalous classes by injecting generic synthetic anomalies into the available data. Moreover, our method can effectively take advantage of all the available information, be it as domain knowledge, or as training labels in the semi-supervised setting. We demonstrate empirically on standard benchmark datasets that our approach obtains a state-of-the-art performance in these settings.

1 Introduction

Detecting anomalies in real-valued time series data has many practical applications, such as monitoring machinery for faults, finding anomalous behavior in IoT sensor data, improving the availability of computer applications and (cloud) infrastructure, and monitoring patients vital signs, among many others. Since Shewhart's pioneering work on statistical process control (Shewhart, 1931), statistical techniques for monitoring and detecting abnormal behavior have been developed, refined, and deployed in countless highly impactful applications.

Recently, deep learning techniques have been successfully applied to various anomaly detection problems (see e.g. the surveys by Ruff et al. (2021) and Pang et al. (2020)). In the particular case of time series, these methods have demonstrated remarkable performance for large-scale monitoring problems such as those encountered by companies like Google (Shipmon et al., 2017), Microsoft (Ren et al., 2019), Alibaba (Gao et al., 2020), and Amazon (Ayed et al., 2020).

Classically, anomaly detection on time series is cast as an unsupervised learning problem, where the training data contains both normal and anomalous instances, but without knowing which is which. However, in many practical applications, a *fully* unsupervised approach can leave valuable information unutilized, as it is often possible to obtain (small amounts of) labeled anomalous instances, or to characterize the relevant anomalies in some general way.

Ideally, an effective method for anomaly detection requires a semi-supervised approach, allowing to utilize information about known anomalous patterns or out-of-distribution observations, if any

[*] Equal contribution.
[†] Work done while working at AWS AI Labs.

Preprint. Under review.

of these are available. Recent developments on deep anomaly detection for computer vision have achieved remarkable performance by following such a learning strategy. A notable example is the line of work leading to the Hypersphere Classifier (Ruff et al., 2018, 2020b,a), which extends the concept of one-class classification to a powerful framework for semi-supervised anomaly detection on complex data.

In this work, we introduce Neural Contextual Anomaly Detection (NCAD), a framework for anomaly detection on time series that can scale seamlessly from the unsupervised to supervised setting, allowing to incorporate additional information, both through labeled examples and through known anomalous patterns. This is achieved by effectively combining recent developments in representation learning for multivariate time series (Franceschi et al., 2019), with a number of deep anomaly detection techniques originally developed for computer vision, such as the Hypersphere Classifier (HSC) (Ruff et al., 2020a) and Outlier Exposure (OE) (Hendrycks et al., 2019), but tailored to the time series setting.

Our approach is based on breaking each time series into overlapping, fixed-size windows. Each window is further divided into two parts: a *context window* and a *suspect window* (see fig. 1), which are mapped into neural representations (embedding) using Temporal Convolutional Networks (TCNs) (Bai et al., 2018). Our aim is to detect anomalies in the *suspect window*. Anomalies are identified in the space of learned latent representations, building on the intuition that anomalies create a substantial perturbation on the embeddings, so when we compare the representation of two overlapping segments, one with the anomaly and one without it, we expect them to be distant.

Time series anomalies are inherently contextual. We account for this in our methodology by extending the HSC loss to a *contextual* hypersphere loss, which dynamically adapts the hypersphere's center based on the context's representation. We use data augmentation techniques to ease the learning of the boundary between the normal and anomalous classes. In particular, we employ a variant of OE to create contextual anomalies, and employ simple injected point outlier anomalies.

In summary, we make the following contributions: **(I)** Propose a simple yet effective framework for time series anomaly detection that achieves state-of-the-art performance across well-known benchmark datasets, covering univariate and multivariate time series, and across the unsupervised, semi-supervised, and fully-supervised settings (Our implementation of NCAD is publicly available [3]); **(II)** Build on related work on deep anomaly detection using the hypersphere classifier (Ruff et al., 2020a) and expand it to introduce contextual hypersphere detection. **(III)** Adapt the Outlier Exposure (Hendrycks et al., 2019) and Mixup (Zhang et al., 2018) methods to the particular case of anomaly detection for time series.

2 Related work

Anomaly Detection (AD) is an important problem with many applications and has consequently been widely studied. We refer the reader to one of the recent reviews in the topic for a general overview of methods (Chandola et al., 2009; Ruff et al., 2021; Pang et al., 2020).

We are interested in anomaly detection for time series. This is a problem typically framed in an unsupervised way. A traditional approach is to use a predictive model, estimating the distribution (or confidence bands) of future values conditioned on historical observations, and mark observations as anomalous if they are considered unlikely under the model (Shipmon et al., 2017). Forecasting models such as ARIMA or exponential smoothing methods are often used here, assuming a Gaussian noise distribution. Siffer et al. (2017) propose SPOT and DSPOT, which detect outliers in time series using extreme value theory to model the tail of the distribution.

New advances on deep learning models for anomaly detection have become popular recently. For time series, some models have maintained the classical predictive approach, and introduced flexible neural networks for the dependency structure, yielding significant improvements. Shipmon et al. (2017) use deep (recurrent) neural networks to parametrize a Gaussian distribution and use the tail probability to detect outliers.

Effective ideas for deep anomaly detection that deviate from the predictive approach have been successfully imported to the time series domain from other fields. Reconstruction based methods, e.g.

[3] `https://github.com/Francois-Aubet/gluon-ts/tree/adding_ncad_to_nursery/src/gluonts/nursery/ncad`

with Variational Auto-Encoders (VAEs), or density based methods , e.g. with Generative Adversarial Networks (GANs): DONUT uses a VAE to predict the distribution of sliding windows; LSTM-VAE (Park et al., 2018) uses a recurrent neural network with a VAE; OMNIANOMALY (Su et al., 2019) extends this framework with deep innovation state space models and normalizing flows; ANOGAN (Schlegl et al., 2017) uses GANs to model sequences of observations and estimate their probabilities in latent space. MSCRED (Zhang et al., 2019) uses convolutional auto-encoders and identifies anomalies by measuring the reconstruction error.

Compression-based approaches have become very popular in image anomaly detection. The working principle is similar to the one-class classification used in the support vector data description method (Tax & Duin, 2004): instances are mapped to latent representations which are pulled together during training, forming a sphere in the latent space; instances that are distant from the center are considered anomalous. (Ruff et al., 2018, 2020b) build on this idea to learn a neural mapping $\phi(\cdot;\theta) : \mathbb{R}^D \to \mathbb{R}^E$, such that the representations of nominal points concentrate around a (fixed) center $\boldsymbol{c}$, while anomalous points are mapped away from that center. In the unsupervised case, DeepSVDD (Ruff et al., 2018) achieves this by minimizing the Euclidean distance $\sum_i ||\phi(\boldsymbol{w}_i;\theta) - \boldsymbol{c}||^2$, subject to a suitable regularization of the mapping and assuming that anomalies are rare.

THOC (Shen et al., 2020) applies this principle to the context of time series, by extending the model to consider multiple spheres to obtain more convenient representations. This method differs from our work in two ways: it relies on a dilated recurrent neural network with skip connections to handle the contextual aspect of the data, we use a much simpler network and use a context window to handle the contextuality. Then, our method can seamlessly handle the semi-supervised setting and benefits from our data augmentation techniques.

Ruff et al. (2020a) propose Hypersphere Classifier (HSC), improving on DeepSVDD by training the network using the standard Binary Cross-Entropy (BCE) loss, this way extending the approach to the (semi-)supervised setting. With this method, they can rely on labeled examples to regularize the training and do not have to resort to limiting the network. In particular, the HSC loss is given by setting the pseudo-probability of an anomalous instance ($y = 1$) as $p = 1 - \ell(\phi(\boldsymbol{w}_i;\theta))$, i.e.

$$-(1-y_i)\log \ell(\phi(\boldsymbol{w}_i;\theta)) - y_i \log(1 - \ell(\phi(\boldsymbol{w}_i;\theta))) \;, \tag{1}$$

where $\ell : \mathbb{R}^E \to [0,1]$ maps the representation to a probabilistic prediction. Choosing $\ell(\boldsymbol{z}) = \exp(-||\boldsymbol{z}||^2)$, leads to a spherical decision boundary in representation space, and reduces to the DeepSVDD loss (with center $\boldsymbol{c} = 0$) when all labels are 0.

Current work on semi-supervised anomaly detection indicates that including even only few labeled anomalies can already yield remarkable performance improvements on complex data (Ruff et al., 2021; Liznerski et al., 2020; Tuluptceva et al., 2020). A powerful resource in this line is Outlier Exposure (OE) (Hendrycks et al., 2019), which improves detection by incorporating large amounts of out-of-distribution examples from auxiliary datasets during training. Despite such negative samples may not coincide with ground-truth anomalies, such contrasting can be beneficial for learning characteristic representations of normal concepts. Moreover, the combination of Outlier Exposure and expressive representations with the hyperphere classifier have shown exceptional results for Deep AD on images (Ruff et al., 2020a; Deecke et al., 2020).

For time series data, however, artificial anomalies and related data augmentation techniques have not been studied extensively. Smolyakov et al. (2019) used artificial anomalies to select thresholds in ensembles of anomaly detection models. Most closely related to our approach, SR-CNN Ren et al. (2019) trains a supervised CNN on top of an unsupervised anomaly detection model (SR), by using labels from injected single point outliers.

Fully supervised methods are not as widely studied because labeling all the anomalies is too expensive and unreliable in most applications. An exception is the work by Liu et al. (2015) who propose a system to continuously collect anomaly labels and to iteratively re-train and deploy a supervised random forest model. The U-NET-DEWA approach of Gao et al. (2020) relies on preprocessing using robust time series decomposition to train a convolutional network in a supervised way, relying on data augmentations that preserve the anomaly labels to increase the training set size.

3 Neural Contextual Anomaly Detection

This section describes our anomaly detection framework and its building blocks. We combine a window-based anomaly detection approach with a flexible training paradigm and effective heuristics for data augmentation to produce a state-of-the-art system for anomaly detection.

We consider the following general time series anomaly detection problem: We are given a collection of N discrete-time time series $\boldsymbol{x}_{1:T_i}^{(i)}$, $i = 1, \ldots, N$ where for time series i and time step $t = 1, \ldots, T_i$ we have an observation vector $\boldsymbol{x}_t^{(i)} \in \mathbb{R}^D$. We further assume that we are given a corresponding, set of partial anomaly labels $y_{1:T_i}^{(i)}$ with $y_t^{(i)} \in \{0, 1, ?\}$, indicating whether the corresponding observation $\boldsymbol{x}_t^{(i)}$ is normal (0), anomalous (1), or unlabeled (?).

The goal is to predict anomaly labels $\hat{y}_{1:T}$, with $y_t \in \{0, 1\}$ given a time series $\boldsymbol{x}_{1:T}$. Instead of predicting the binary labels directly, we predict a positive anomaly score for each time step, which can subsequently be thresholded to obtain anomaly labels satisfying a desired precision/recall trade-off.

3.1 Window-based Contextual Hypersphere Detection

Similar to other work on time series AD (e.g. Ren et al. (2019); Guha et al. (2016)), we convert the time series problem to a vector problem by splitting each time series into a sequence of overlapping, fixed-size windows $\boldsymbol{w}$ of length L. A key element of our approach is that within each window, we identify two segments: a *context window* $\boldsymbol{w}^{(c)}$ of length C and *suspect window* of length S: $\boldsymbol{w} = (\boldsymbol{w}^{(c)}, \boldsymbol{w}^{(s)})$, where we typically choose $C \gg S$. Our goal is to detect anomalies in the suspect window relative to the local context provided by the context window. This split not only naturally aligns with the typically contextual nature of anomalies in time series data, it also allows for short suspect windows (even $S = 1$), minimizing detection delays and improving anomaly localization.

Intuitively, our approach is based on the idea that we can identify anomalies by comparing representation vectors $\phi(\boldsymbol{w}; \theta)$ and $\phi(\boldsymbol{w}^{(c)}; \theta)$, obtained by applying a neural network feature extractor $\phi(\cdot; \theta)$, which is trained in such a way that representations are pulled together if there is no anomaly present in the suspect window $\boldsymbol{w}^{(s)}$, and pushed apart otherwise.

We propose a loss function, which can be seen as *contextual* version of the Hypersphere Classifier (equation 1) by considering a loss function which contrasts the representation of the context window with the representation of the full window:

$$-(1 - y_i) \log \left(\ell \left(\operatorname{dist}(\phi(\boldsymbol{w}_i; \theta), \phi(\boldsymbol{w}_i^{(c)}; \theta) \right) \right) - y_i \log \left(1 - \ell \left(\operatorname{dist}(\phi(\boldsymbol{w}_i; \theta), \phi(\boldsymbol{w}_i^{(c)}; \theta) \right) \right) .$$

In our experiments we follow Ruff et al. (2020a) and use the Euclidean distance $\operatorname{dist}(x, z) = \|x - z\|_2$ and a radial basis function $\ell(z) := \exp(-z^2)$, to create a spherical decision boundary as in HSC/DeepSVDD, resulting in the loss function

$$(1 - y_i) \left\| \phi(\boldsymbol{w}_i; \theta) - \phi(\boldsymbol{w}_i^{(c)}; \theta) \right\|_2^2 - y_i \log \left(1 - \exp \left(- \left\| \phi(\boldsymbol{w}_i; \theta) - \phi(\boldsymbol{w}_i^{(c)}; \theta) \right\|_2^2 \right) \right) . \quad (2)$$

Intuitively, this is the HSC loss where the center $\boldsymbol{c}$ of the hypersphere is chosen dynamically for each instance as the representation of the context. This introduces an inductive bias: representations of the context window and representations of the full window should be different if an anomaly occurs in the suspect window. As we show in our empirical analysis, this inductive bias makes the model more label efficient leads to better generalization. In particular, we show that when this model is trained using generic injected anomalies such as point outliers, it is able to generalize to the more complex anomalies found in real world datasets.

3.2 NCAD architecture & training

Our model identifies anomalies in a space of learned latent representations, building on the intuition that: if an anomaly is present in the suspect window $\boldsymbol{w}^{(s)}$, then representation vectors constructed from $\boldsymbol{w}$ and $\boldsymbol{w}^{(c)}$ should be distant.

As illustrated in fig. 1, our NCAD architecture has three components:

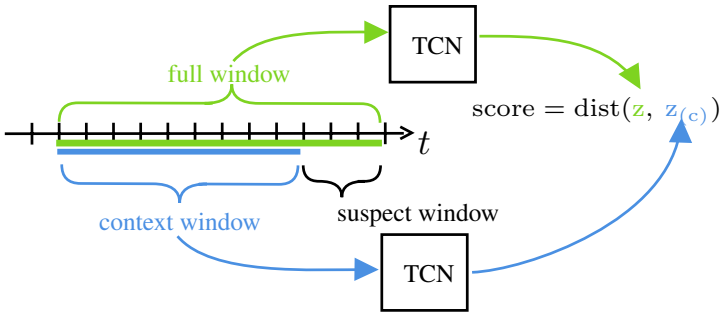


Figure 1: NCAD encodes two windows that differ by a suspect window using the same TCN network and computes a distance score of the embeddings. The model is trained to give a high score for instances with an anomaly in the suspect window.

1. A neural network *encoder* $\phi(\cdot;\theta)$, that maps input sequences to representation vectors in $\mathbb{R}^E$. The same encoder is applied both to the full window and to the context window, resulting in representations $z = \phi(\boldsymbol{w};\theta)$ and $z_{(c)} = \phi(\boldsymbol{w}_{(c)};\theta)$, respectively. While any neural network could be used, in our implementation we opt for a CNN with exponentially dilated causal convolutions (van den Oord et al., 2016), in particular the TCN architecture (Bai et al., 2018; Franceschi et al., 2019) with adaptive max-pooling along the time dimension.
2. A distance-like function, $\mathrm{dist}(\cdot,\cdot) : \mathbb{R}^E \times \mathbb{R}^E \to \mathbb{R}^+$, to compute the similarity between the representations z and $z_{(c)}$.
3. A probabilistic scoring function $\ell(z) = \exp(-|z|)$, which creates a spherical decision boundary centered at the embedding of the context window.

The parameters θ of the encoder are learned by minimizing the classification loss on minibatches of windows, $\boldsymbol{w}$. These are sampled uniformly at random (across time series and across time) from the training data set $\{\boldsymbol{x}^{(i)}_{1:T_i}\}$ after applying the data augmentation techniques that follow.

Rolling predictions While our window based approach allows the model to decide *if* an anomaly is present in the suspect window, in many applications it is important to react quickly when an anomaly occurs, or to locate the anomaly with some accuracy. To support these requirements, we apply the model on rolling windows of the time series. Each time point can then be part of different suspect windows corresponding to different rolling windows had so is given multiple anomaly scores. Using these we can either alert on the first high score, to reduce time to alert, or average the scores for each point to pin-point the anomalies in time more accuratly.

3.3 Data augmentation

In addition to the contrastive classifier, we utilize a collection of data augmentation methods that inject synthetic anomalies, allowing us to rely on a supervised training objective without requiring ground-truth labels. While can rely on the hypersphere to train without any labels, having some anomalous examples labels allow to greatly improve the performance, as it has been observed in computer vision (Hendrycks et al., 2019). We cannot effectively rely on ground-truth anomalies as few datasets have any and in practice it is very costly to obtain training labels; therefore, we propose to generate the anomalous example. These data augmentation methods explicitly do *not* attempt to characterise the full data distribution of anomalies, which would be infeasible; rather, we combine effective generic heuristics that work well for detecting common types of out-of-distribution examples.

Contextual Outlier Exposure (COE) — Motivated by the success of Outlier Exposure for *out-of-distribution* detection Hendrycks et al. (2019), we propose a simple task-agnostic method to create contextual out-of-distribution examples. Given a data window $\boldsymbol{w} = (\boldsymbol{w}^{(c)}, \boldsymbol{w}^{(s)})$, we induce anomalies into the suspect segment, $\boldsymbol{w}^{(s)}$, by replacing a chunk of its values with values taken from another time series. The replaced values in $\boldsymbol{w}^{(s)}$ will most likely break the temporal relation with their neighboring context, therefore creating an out of distribution example. In our implementation,

we apply COE at training time by selecting random examples in a minibatch and permuting a random length of their suspect windows (visualizations in appendix B.1). In multivariate time series, as anomalies do not have to happen in all dimensions, we randomly select a subset of the dimensions in which the windows are swapped.

Anomaly Injection — We propose to inject simple single **Point Outliers (po)** in the time series. We use a simple method: at a set of randomly selected time points we add (or subtract) a spike to the time series. The spike is proportional to the inter-quartile range of the points surrounding the spike location. Like for COE, in multivariate time series we simply select a random subset of dimensions on which we add the spike. These simple point outliers serve the same purpose as COE: create clear labeled abnormal points to help the learning of the hypersphere. (visualizations in appendix B.2).

In addition to these, in some practical applications, it is possible to identify general characteristics of anomalies that should be detected. Some widely-known anomalous patterns include: sudden changes in the location or scale of the series (change-points); interruption of seasonality, etc. We have used this approach in our practical application and the domain knowledge allowed to improve the detection performance. As they require and domain knowledge it would be unfair to compare our method when incorporating these; therefore, in the results table we only use the point outliers described above.

Window Mixup — If we do not have access to training labels and know little about the relevant anomalies, we can only rely on COE and po, which may result in significantly missmatch between injected and true anomalies. To improve generalization of our model in this case, we propose to create linear combinations of training examples inspired by the MIXUP procedure Zhang et al. (2018).

MIXUP was proposed in the context of computer vision and creates new training examples out of original samples by using a convex combinations of the features and their labels. This data augmentation technique creates more variety in training examples, but more importantly, the soft labels result in smoother decision functions that generalize better. MIXUP is suited for time series applications: convex combinations of time series most often result in realistic and plausible new time series (see visualizations in appendix B.3). We show that MIXUP can improve generalization of our model even in cases with a large mismatch between injected and true anomalies.

4 Experiments

In this section, we compare the performance of our approach with alternative methods on public benchmark datasets, and exploring the model behavior under different data settings and model variations in ablation studies. Further details on the experiments are included in the supplement.

4.1 Benchmark datasets

We benchmark our method to others on six datasets (more details in appendix E):[4]

Soil Moisture Active Passive satellite (SMAP) and **Mars Science Laboratory rover (MSL)** — Two datasets published by NASA Hundman et al. (2018), with 55 and 27 series respectively. The lengths of the time series vary from 300 to 8500 observations.

Secure Water Treatment (SWaT) — The dataset was collected on a water treatment testbed over 11 days, 36 attacks were launched in the last 4 days and compose the test set. (Mathur & Tippenhauer, 2016) To compare our numbers with Shen et al. (2020), we use the first half of the proposed test set for validation and the second one for test.

Server Machine Dataset (SMD) — Is a 5 weeks long dataset with 28 38-dimensional time series each collected from a different machine in large internet companies (Su et al., 2019).

SMAP, MSL, SWaT, and SMD, each have a pre-defined train/test split, where anomalies in the test set are labeled, while the training set contains unlabeled anomalies.

[4]While we share many of the concerns expressed by Wu & Keogh (2020) about the lack of quality benchmark datasets for time series anomaly detection, we use these commonly-used benchmark datasets here for lack of better alternatives and to enable direct comparison of our approach to competing methods.

YAHOO — A dataset published by Yahoo labs,[5] consisting of 367 real and synthetic time series. Following (Ren et al., 2019), we use the last 50% of the time points of each of the time series as test set and split the rest in 30% training and 20% validation set.

KPI — A univariate dataset released in the AIOPS data competition (kpi). It consists of KPI curves from different internet companies in 1 minute interval. Like (Ren et al., 2019), we use 30% of the train set for validation. For KPI and YAHOO labels are available for all the series.

4.2 Evaluation setup

Measuring the performance of time series anomaly detection methods in a universal way is challenging, as different applications often require different trade-offs between sensitivity, specificity, and temporal localization. To account for this, various measures that improve upon simple point-wise classification metrics have been proposed, e.g. the flexible segment-based score proposed by Tatbul et al. (2018) or the score used in the Numenta anomaly benchmark (Lavin & Ahmad, 2015). To make our results directly comparable, we follow the procedure proposed by Xu et al. (2018) (and subsequently used in other work Su et al. (2019); Ren et al. (2019); Shen et al. (2020)), which offers a practical compromise: point-wise scores are used, but the predicted labels are expanded to mark an entire true anomalous segment as detected correctly if at least one time point was detected by the model.[6] We align our experimental protocol with this body of prior work and report $F1$ scores computed by choosing the best threshold on the test set. For each dataset, the best threshold is chosen and used on all the time series of the test set.

In many real world scenarios, one is interested in detecting anomalies in a streaming fashion on a variety of different time series that may not be known at deployment time. We incorporate these requirements by training a single model on all the training time series of each dataset, and evaluate that model on all the test set time series. Further we use short suspect windows allowing to decide if a point is anomalous or not when it is first observed. We report the performance of this harder detection setting.

Hyperparameters were chosen in the following way: for YAHOO, KPI and SWaT, as the validation datasets have anomaly labels available, we use a Bayesian optimization (Perrone et al., 2020) for parameter tuning, by maximizing the F1 score on the validation set. If no validation set with labels is available, we use a set of standard hyperparameter settings inferred from the datasets with validation datasets. (see details in the supplement). On each dataset we pick the context window length to roughly match the length of the seasonal patterns in the time series.

We run the model 10 times on each of the benchmark datasets and report mean and standard deviation. We use standard AWS EC2 ml.p3.2xlarge instances with a single-core Tesla V100 GPU. Training the model on one of the benchmark datasets takes on average 90 minutes. In our code [7] we provide scripts to reproduce the results on the benchmark datasets shown below.

4.3 Benchmark results

Table 1 shows the performance of our NCAD approach compared against the state-of-the-art methods on two commonly used univariate datasets. As these datasets contains labels for anomalies both on the training and the test set, we evaluate our method on them both in the supervised setting (*(sup.)*) and the unsupervised setting (*(un.)*) We take the numbers from Ren et al. (2019). Our approach significantly outperforms competing approaches on YAHOO, performs similarly to the best unsupervised approach on KPI, and slightly worse than the best supervised approach. It is important to note that while other methods are either designed for the supervised or unsupervised setting, our method can be used seamlessly in both settings.

The YAHOO-supervised experiments are included in appendix C.2, where we compare against the supervised approach of (Gao et al., 2020), which represents the state-of-the-art in this setting to the best of our knowledge. Our approach outperforms their approach significantly with 79% point-wise F1 score versus 69.3% F1 score for their approach.

[5] `https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70`

[6] We use the implementation by Su et al. (2019): `https://github.com/NetManAIOps/OmniAnomaly/`.

[7] `https://github.com/Francois-Aubet/gluon-ts/tree/adding_ncad_to_nursery/src/gluonts/nursery/ncad`

Table 1: F1 score of the model on univariate datasets

Model	YAHOO (un.)	KPI (un.)	KPI (sup.)
SPOT (Siffer et al., 2017)	33.8	21.7	—
DSPOT (Siffer et al., 2017)	31.6	52.1	—
DONUT (Xu et al., 2018)	2.6	34.7	—
SR (Ren et al., 2019)	56.3	62.2	—
SR-CNN (Ren et al., 2019)	65.2	**77.1**	—
SR+DNN (Ren et al., 2019)	—	—	**81.1**
NCAD w/ COE, po , mixup	**81.16 ± 1.43**	**76.64 ± 0.89**	79.20 ± 0.92

Table 2: F1 score of the model on multivariate datasets

Model	SMAP	MSL	SWaT	SMD
AnoGAN (Schlegl et al., 2017)	74.59	86.39	86.64	—
DeepSVDD (Ruff et al., 2018)	71.71	88.12	82.82	—
DAGMM (Zong et al., 2018)	82.04	86.08	85.38	70.94
LSTM-VAE (Park et al., 2018)	75.73	73.79	86.39	78.42
MSCRED (Zhang et al., 2019)	77.45	85.97	86.84	—
OmniAnomaly (Su et al., 2019)	84.34	89.89	—	**88.57**
MTAD-GAT (Zhao et al., 2020)	90.13	90.84	—	—
THOC (Shen et al., 2020)	**95.18**	93.67	88.09	—
NCAD w/ COE, po , mixup	**94.45 ± 0.68**	**95.60 ± 0.59**	**95.28 ± 0.76**	80.16 ± 0.69

Table 2 shows the performance of our NCAD approach compared against the state-of-the-art methods. None of these datasets provides labels for the anomalies in the training set, all benchmark methods are designed for unsupervised anomaly detection. Our method outperforms THOC by a reasonable margin both on MSL and SWaT. On SMAP while our average score is slightly lower, the difference is within the variance. OmniAnomaly (Su et al., 2019) is the state of the art on SMD, our numbers are only second to theirs. We note that OmniAnomaly is considerably more costly and less scalable, since it trains one model for each of the 28 time series of the dataset, while we train a single global model.

4.4 Ablation study

To better understand the advantage brought by each of the components of our method, we perform an ablation study on the SMAP and MSL datasets, shown in fig. 2a. We average two runs for each configuration, the full table with all configurations and standard deviation is shown and discussed in appendix C.1. The row labeled "- contextual ..." does not use the contextual hypersphere described in section 3.1, but instead a model trained using the original hypersphere classifier loss on the whole-window representation $\phi(\boldsymbol{w};\theta)$. The contextual loss function provides a substantial performance boost, making our approach competitive even without the data augmentation techniques. Each of the data augmentation techniques improves the performance further. A further ablation study on the supervised Yahoo dataset can be found in table 4.

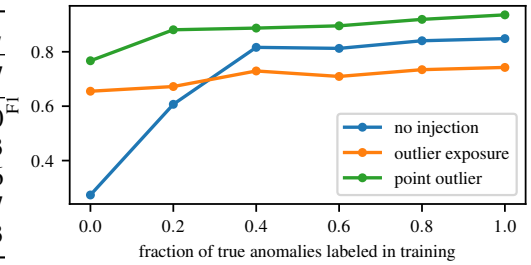
4.5 Scaling from unsupervised to supervised

To investigate how the performance of our approach changes as we scale from unsupervised, to semi-supervised, to fully supervised, we measure the performance of our approach as a function of the amount of ground truth labels on the YAHOO dataset, shown in fig. 2b. Firstly, we observe that the performance increases steadily with the amount of true anomaly labels, as desired. Secondly, by using synthetic anomalies (either po or COE), we can significantly boost the performance in the regime when no or only few labels are available. Finally, by using an injection technique that is well-aligned with the desired type of anomalies (po in this case, as YAHOO contains a large number of single-point outliers), one can significantly improve performance over relying solely on the labeled data, this is explained by the very high class imbalance in anomaly detection. The flipside is, of

(a) Ablation study on SMAP and MSL

Model	SMAP	MSL
THOC (Shen et al., 2020)	95.18	93.67
NCAD w/ COE, po , mixup	94.45	95.60
- po	94.28	94.73
- COE	88.59	94.66
- mixup - COE - po	66.9	79.47
- contextual - mixup - COE - po	55.09	36.03

(b) F1 score of NCAD on the YAHOO dataset trained with only a fraction of training anomalies being labeled.



course, that injecting anomalies that may be significantly different from the desired anomalies (COE in this case) can ultimately hurt when enough labeled examples are available.

4.6 Using specialized anomaly injection methods

While in all our benchmarks we rely on completely generic anomalies for injection (COE and po), a by-product of our methodology is that the model can be guided towards detecting the desired class of anomalies by designing anomaly injection methods that mimic the true anomalies. Designing such methods is often simple compared to finding enough examples of true anomalies as they are rare. Figure 3a demonstrates the effectiveness of this approach: The first dimension of the SMAP dataset contains slow slopes that are labeled as anomalous in the dataset. These are harder to detect for our model when only using COE and po because these cannot create similar behavior. We can design a simple anomaly injection that injects slopes to randomly selected region and labels it as anomalous. Training NCAD with these slopes gives a model that achieves a much better score.

This approach can be effective in applications where anomalies are subtle and closer to the normal data, and where some prior knowledge is available about the kind of anomalies that are to be detected. However one may not have this prior knowledge or the resources required to create these injections. This is a limitation of this technique which prevents it from being generally applicable. This is the reason why we did not use in for the comparison to the other methods.

(b) F1 score vs. width of true anomalies for models trained only on point outliers, with different fractions of training examples mixed-up.

(a) F1 score on the Performance first dimension of SMAP with specialized anomaly injections.

Model	SMAP 1st dimension
NCAD	93.38
NCAD + injections	96.48

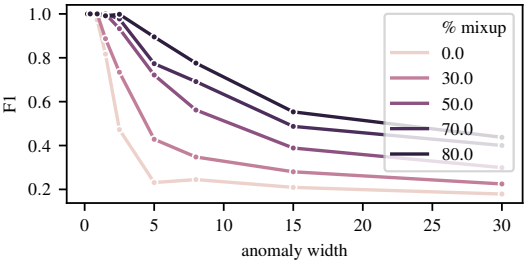


Figure 3: Investigating the potential of anomaly injections taking advantage of domain knowledge, and investigating the generalization of the model form po when this knowledge is not available.

4.7 Generalization from injected anomalies

Artificial anomalies will always differ from the true anomalies to some extent, be it the ones created by COE, po, or more complex methods. This requires the model to bridge this gap and generalize from imperfect training examples to true anomalies. By design the hypersphere formulation can help to bridge the gap, and we use MIXUP further improve the generalization capabilities of the model. Figure 3b shows the results of an experiment exploring one aspect of this generalization ability for NCAD. The model is trained with injected single-point outliers, and we measure the detection performance for anomalies of longer width. For this experiment we use a synthetic base data set

containing simple sinusoid time series with Gaussian noise. We create multiple datasets from this base dataset adding true anomalies of varying width by convolving spike anomalies with Gaussian filters of different widths. For training, regardless of the shape of the true anomalies, we use po and train models using different MIXUP rates, i.e., fraction of training examples with MIXUP applied. We observe that MIXUP helps the model to generalize in this setting:the higher the MIXUP rate, the better the model generalizes to anomalies that differ from the injected examples, achieving higher F1 scores.

5 Discussion

We present NCAD, a methodology for anomaly detection in time series that achieves state-of-the-art performance in a broad range of settings, including both the univariate and multivariate cases, as well as across the unsupervised, semi-supervised, and supervised anomaly detection regimes. We demonstrate that combining expressive neural representation for time series with data augmentation techniques can outperform traditional approaches such as predictive models or methods based on reconstruction error.

We do not foresee clear potential negative societal impact of this work. Time series anomaly detection is a general problem which is applied in many different domains, such as cyber-security where it can be used to automatically prevent attacks to power plants or hospitals. While the anomaly detection results of our approach are good, we think that the detection of the algorithm should not be blindly followed in medical application impacting directly the patients health.

References

Aiops challenge. `http://iops.ai/dataset_detail/?id=10`.

Ayed, F., Stella, L., Januschowski, T., and Gasthaus, J. Anomaly detection at scale: The case for deep distributional time series models. *arXiv preprint arXiv:2007.15541*, 2020.

Bai, S., Kolter, J. Z., and Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv preprint arXiv:1803.01271*, mar 2018. URL `http://arxiv.org/abs/1803.01271`.

Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

Deecke, L., Ruff, L., Vandermeulen, R. A., and Bilen, H. Deep Anomaly Detection by Residual Adaptation. *arXiv preprint arXiv:2010.02310*, oct 2020. URL `http://arxiv.org/abs/2010.02310`.

Falcon, W. A. PyTorch Lightning, 2019. URL `https://github.com/PyTorchLightning/pytorch-lightning`.

Franceschi, J. Y., Dieuleveut, A., and Jaggi, M. Unsupervised scalable representation learning for multivariate time series. In *Proceedings of the 33rd Conference on Neural Information Processing Systems, NeurIPS 2019*, volume 32, jan 2019.

Gao, J., Song, X., Wen, Q., Wang, P., Sun, L., and Xu, H. RobustTAD: Robust Time Series Anomaly Detection via Decomposition and Convolutional Neural Networks. *arXiv preprint arXiv:2002.09545*, feb 2020. URL `http://arxiv.org/abs/2002.09545`.

Guha, S., Mishra, N., Roy, G., and Schrijvers, O. Robust random cut forest based anomaly detection on streams. In *International conference on machine learning*, pp. 2712–2721. PMLR, 2016.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

Hendrycks, D., Mazeika, M., and Dietterich, T. Deep anomaly detection with outlier exposure. In *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019*, dec 2019. URL http://arxiv.org/abs/1812.04606.

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

Lavin, A. and Ahmad, S. Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 38–44. IEEE, 2015.

Liberty, E., Karnin, Z., Xiang, B., Rouesnel, L., Coskun, B., Nallapati, R., Delgado, J., Sadoughi, A., Astashonok, Y., Das, P., Balioglu, C., Chakravarty, S., Jha, M., Gautier, P., Arpin, D., Januschowski, T., Flunkert, V., Wang, Y., Gasthaus, J., Stella, L., Rangapuram, S., Salinas, D., Schelter, S., and Smola, A. Elastic machine learning algorithms in amazon sagemaker. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 731–737, 2020.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollar, P. Focal Loss for Dense Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, feb 2020. ISSN 0162-8828. doi: 10.1109/TPAMI.2018.2858826. URL http://arxiv.org/abs/1708.02002https://ieeexplore.ieee.org/document/8417976/.

Liu, D., Zhao, Y., Xu, H., Sun, Y., Pei, D., Luo, J., Jing, X., and Feng, M. Opprentice: Towards practical and automatic anomaly detection through machine learning. In *Proceedings of the 2015 Internet Measurement Conference*, pp. 211–224, 2015.

Liznerski, P., Ruff, L., Vandermeulen, R. A., Franks, B. J., Kloft, M., and Müller, K.-R. Explainable Deep One-Class Classification. In *Proceedings of the 9th International Conference on Learning Representations, ICLR 2021*, jul 2020. URL https://github.com/liznerski/fcddhttp://arxiv.org/abs/2007.01760.

Mathur, A. P. and Tippenhauer, N. O. Swat: a water treatment testbed for research and training on ics security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, pp. 31–36. IEEE, 2016.

Pang, G., Shen, C., Cao, L., and Hengel, A. v. d. Deep learning for anomaly detection: A review. *arXiv preprint arXiv:2007.02500*, 2020.

Park, D., Hoshi, Y., and Kemp, C. C. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters*, 3(3):1544–1551, 2018.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Proceedings of the 33rd Conference on Neural Information Processing Systems, NeurIPS 2019*, dec 2019. URL http://arxiv.org/abs/1912.01703.

Perrone, V., Shen, H., Zolic, A., Shcherbatyi, I., Ahmed, A., Bansal, T., Donini, M., Winkelmolen, F., Jenatton, R., Faddoul, J. B., Pogorzelska, B., Miladinovic, M., Kenthapadi, K., Seeger, M., and Archambeau, C. Amazon SageMaker Automatic Model Tuning: Scalable Black-box Optimization. Technical report, Amazon, dec 2020. URL http://arxiv.org/abs/2012.08489.

Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X.-A., Xing, T., Yang, M., Tong, J., and Zhang, Q. Time-Series Anomaly Detection Service at Microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, volume 19, pp. 3009–3017, New York, NY, USA, jul 2019. ACM. ISBN 9781450362016. doi: 10.1145/3292500.3330680. URL https://doi.org/10.1145/3292500.3330680https://dl.acm.org/doi/10.1145/3292500.3330680.

Ruff, L., Vandermeulen, R. A., Görnitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Uller, E. M., and Kloft, M. Deep One-Class Classification. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, pp. 4393—-4402, 2018. URL http://proceedings.mlr.press/v80/ruff18a.

Ruff, L., Vandermeulen, R. A., Franks, B. J., Müller, K.-R., and Kloft, M. Rethinking Assumptions in Deep Anomaly Detection. *arXiv preprint arXiv:2006.00339*, may 2020a. URL http://arxiv.org/abs/2006.00339.

Ruff, L., Vandermeulen, R. A., Görnitz, N., Binder, A., Müller, E., Müller, K.-R., and Kloft, M. Deep Semi-Supervised Anomaly Detection. In *Proceedings of the 8th International Conference on Learning Representations, ICLR 2020*, jun 2020b. URL http://arxiv.org/abs/1906.02694.

Ruff, L., Kauffmann, J. R., Vandermeulen, R. A., Montavon, G., Samek, W., Kloft, M., Dietterich, T. G., and Muller, K.-R. A Unifying Review of Deep and Shallow Anomaly Detection. *Proceedings of the IEEE*, 109(5):756–795, may 2021. ISSN 0018-9219. doi: 10.1109/JPROC.2021.3052449. URL https://www.statista.com/http://arxiv.org/abs/2009.11732https://ieeexplore.ieee.org/document/9347460/.

Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., and Langs, G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging*, pp. 146–157. Springer, 2017.

Shen, L., Li, Z., and Kwok, J. Timeseries anomaly detection using temporal hierarchical one-class network. *Advances in Neural Information Processing Systems*, 33, 2020.

Shewhart, W. A. *Economic control of quality of manufactured product*. Macmillan And Co Ltd, London, 1931.

Shipmon, D. T., Gurevitch, J. M., Piselli, P. M., and Edwards, S. T. Time series anomaly detection; detection of anomalous drops with limited features and sparse examples in noisy highly periodic data. *arXiv preprint arXiv:1708.03665*, 2017.

Siffer, A., Fouque, P.-A., Termier, A., and Largouet, C. Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1067–1075, 2017.

Smolyakov, D., Sviridenko, N., Ishimtsev, V., Burikov, E., and Burnaev, E. Learning Ensembles of Anomaly Detectors on Synthetic Data. *arXiv:1905.07892 [cs, stat]*, May 2019.

Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., and Pei, D. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2828–2837, 2019.

Tatbul, N., Lee, T. J., Zdonik, S., Alam, M., and Gottschlich, J. Precision and recall for time series. *Advances in Neural Information Processing Systems*, 31:1920–1930, 2018.

Tax, D. M. and Duin, R. P. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

Tuluptceva, N., Bakker, B., Fedulova, I., Schulz, H., and Dylov, D. V. Anomaly Detection with Deep Perceptual Autoencoders. *arXiv preprint arXiv:2006.13265*, jun 2020. URL http://arxiv.org/abs/2006.13265.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. WaveNet: A Generative Model for Raw Audio. Technical report, Google, sep 2016. URL http://arxiv.org/abs/1609.03499https://research.google/pubs/pub45774/.

Wu, R. and Keogh, E. J. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *arXiv preprint arXiv:2009.13807*, 2020.

Xu, H., Chen, W., Zhao, N., Li, Z., Bu, J., Li, Z., Liu, Y., Zhao, Y., Pei, D., Feng, Y., et al. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 World Wide Web Conference*, pp. 187–196, 2018.

Zaheer, M., Reddi, S. J., Sachan, D., Kale, S., and Kumar, S. Adaptive methods for nonconvex optimization. In *Proceedings of the 32nd Conference on Neural Information Processing Systems, NIPS 2018*, volume 2018-Decem, pp. 9793–9803, 2018.

Zhang, C., Song, D., Chen, Y., Feng, X., Lumezanu, C., Cheng, W., Ni, J., Zong, B., Chen, H., and Chawla, N. V. A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:1409–1416, jul 2019. ISSN 2374-3468. doi: 10.1609/aaai.v33i01.33011409. URL `www.aaai.orghttps://aaai.org/ojs/index.php/AAAI/article/view/3942`.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. Mixup: Beyond Empirical Risk Minimization. In *Proceedings of the Sixth International Conference on Learning Representations, ICLR 2018*, oct 2018.

Zhao, H., Wang, Y., Duan, J., Huang, C., Cao, D., Tong, Y., Xu, B., Bai, J., Tong, J., and Zhang, Q. Multivariate time-series anomaly detection via graph attention network. *arXiv preprint arXiv:2009.02040*, 2020.

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., and Chen, H. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.

A Model Architecture

A.1 Encoder

Our encoder function $g(\cdot) : \mathbb{R}^{D \times L} \to \mathbb{R}^{E}$ [8] is similar to the encoder proposed by Franceschi et al. (2019) for generating universal representations of multivariate time series.

The architecture is based on multi-stack Temporal Convolutional Networks (TCNs) (Bai et al., 2018), which combines causal convolutions with residual connections. The output of this causal network is passed to an adaptive max pooling layer, aggregating the temporal dimension into a fixed-size vector.[9] A linear transformation is applied to produce the unnormalized vector representations, which are then $L2$-normalized to produce the final output of the encoder. See fig. 4 for an illustration.

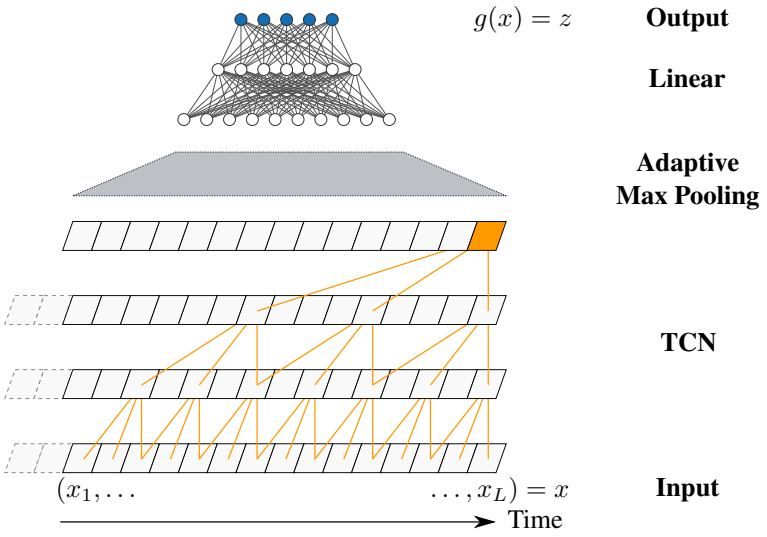


Figure 4: Illustration of the encoder architecture.

A.2 Distance

We introduce a "distance" [10] function, $\text{dist}(\cdot, \cdot) : \mathbb{R}^E \times \mathbb{R}^E \to \mathbb{R}^+$, as a central element in our approach. For a given window $x = (x_{(c)}, x_{(s)})$, the distance function is used to compare the embeddings of the entire window, $z = \phi(x; \theta)$, with the embedding of its corresponding context segment, $z_{(c)} = \phi(x_{(c)}; \theta)$. The output of this function can be directly interpreted as an anomaly score for the associated suspect segment $x_{(s)}$.

We explored two types of distances: the *Euclidean distance*,

$$\text{dist}_{L2}(x, y) = \|x - y\|_2 = \sqrt{(x - y) \cdot (x - y)},$$

and the *Cosine distance*, defined as a simple logarithmic mapping of the cosine similarity to the positive reals,

$$\text{dist}_{\cos}(x, y) = -\log\left(\frac{1 + \text{sim}(x, y)}{2}\right)$$

where $\text{sim}(x, y) = \dfrac{x \cdot y}{\|x\|_2 \cdot \|y\|_2}$ is the cosine similarity between x and y.

In our experiments, we found that both distances were able to achieve state-of-the-art results in most of the reported benchmarks datasets, with slightly better performance from the Euclidean distance. All the results of NCAD reported in section 4 and appendix C are based on the Euclidean distance.

[8] D denotes the dimension of the time series ($D = 1$ for univariate series), L is the length of the windows, and E the dimension of the vector representation.

[9] In our experiments, using adaptive pooling consistently outperformed the global pooling alternative.

[10] The function $\text{dist}(\cdot, \cdot)$ in our framework is not strictly a distance in the mathematical sense. Specifically, the triangle inequality is not required, but it is expected to be symmetric and non-negative.

Moreover, the NCAD framework can be extended to use other distances, e.g. other L_p norms, the pseudo-Hubber norm used in the Ruff et al. (2020a), or even trainable neural-based distances.

A.3 Probabilistic scoring function and Classification Loss

The final element in our model is the binary classification loss which measures the agreement between target labels y and the assigned anomaly scores.

As described in section 3.2, for a given window $x = (x_{(c)}, x_{(s)})$ and encoder $\phi(\cdot;\theta)$, we compute the anomaly score for the suspect window $x_{(s)}$, as the distance between the corresponding representations of the full window and the context window: $dist(\phi(x;\theta), \phi(x_{(c)};\theta))$.

This score is mapped into a pseudo-probability of an anomaly ($y = 1$) via the probabilistic scoring function $\ell(\cdot)$,

$$p = 1 - \ell(dist(g(x), g(x_{(c)}))),$$

which is used within the Binary Cross-Entropy loss to define the target to minimize during training. We train the encoder $\phi(\cdot;\theta)$ using mini-batch gradient descent (see appendix D), taking B randomly selected windows, and minimizing

$$L_{BCE} = \frac{1}{B}\sum_{i=1}^{B} -\left[y_i \ \log p_i + (1-y_i) \ \log(1-p_i)\right] \ .$$

Alternative classification losses can be considered as an extension of the standard NCAD framework. For example, the Mean Absolute Error (MAE), the Mean Squared Error (MSE), or the Focal Loss (Lin et al., 2020). These losses may be particularly useful in applications with significant contamination of labels.

B Data Augmentation

As presented in section 3.3, our framework relies on three data augmentation methods for time series, we provide more details and some visualizations in this section.

Figure 5 shows four time series drawn from the SMAP benchmark dataset, we use these to visualize each of the data augmentation method.

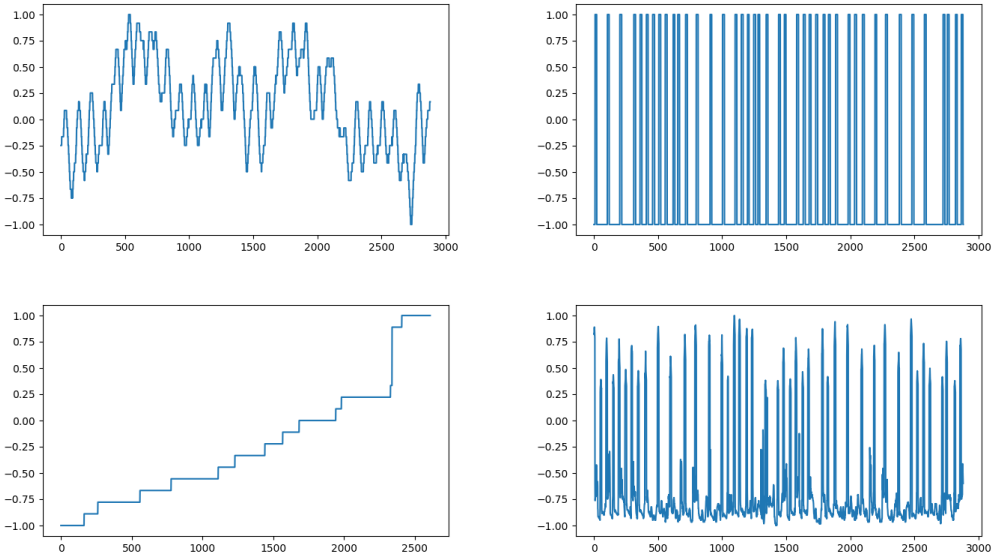


Figure 5: The four original time series used for visualization of the data augmentation methods.

B.1 Contextual Outlier Exposure (COE)

As described in section 3.3, we use COE at training time to generate additional anomalous training examples. These are created by replacing a chunk of values in one suspect window with values taken from another suspect window in the same batch.

As an example, consider fig. 6, each time series has the window between 1500 and 1550 swapped with its horizontal neighbor. We can see that this creates an anomalous window that does not follow the expected behavior. In some cases, the swapping of values create jumps, as in (c); in other cases the change is more subtle, like in (d), where the series becomes constant for the duration of the interval or (a) and (b) where the regular pattern is broken. [11]

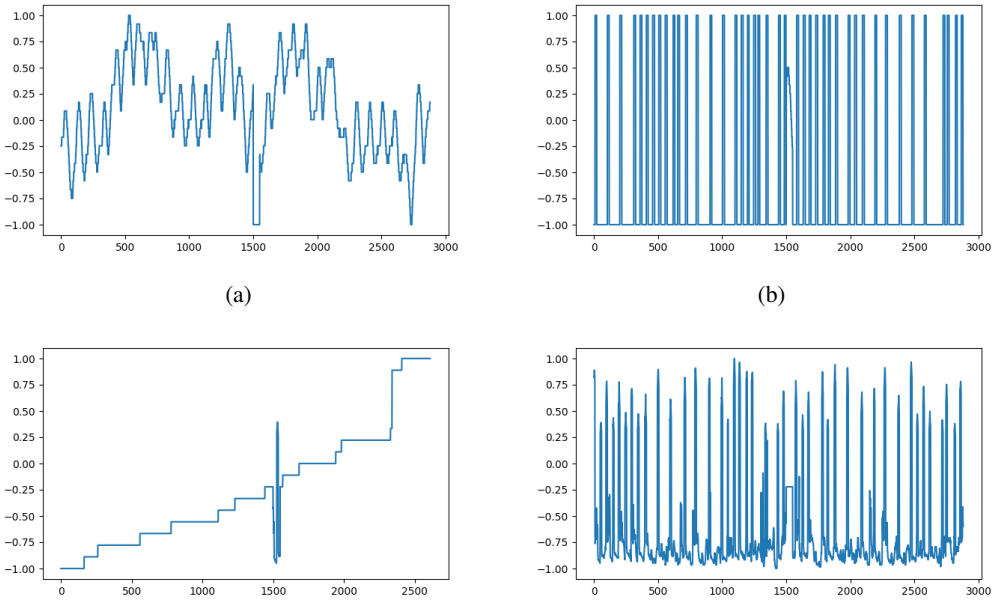


Figure 6: Visualization of coe. Each of the series has its window between 1500 and 1550 swapped with its horizontal neighbor time series: (a) swaps with (b) and (c) swaps with (b)).

B.2 Point Outliers (PO)

As described in section 3.3, Point Outliers allow to add single isolated outliers to the time series. We simply add a spike to the time series at a random location in time. By default, the spike is between 0.5 and 3 time the inter-quartile range of the 100 points around the spike location.

With this method, the injected spike can be a local outlier, but is not necessarily a global outlier as its magnitude could be within the range of other values in the time series. Similarly to COE, in the case of multivariate time series we select a random subset of dimensions on which we add the spike. In fig. 7 we visualizes some examples of the injected point outliers. These are added to the 1550th value of each of the time series. We can see that they break the normal patterns but do not necessarily result in extreme events. [12]

[11] Our implementation of COE can be found in file `src/ncad/model/outlier_exposure.py` of the supplementary code.

[12] Our implementation of po can be found in file `src/ncad/ts/transforms/spikes_injection.py` of the supplementary code.

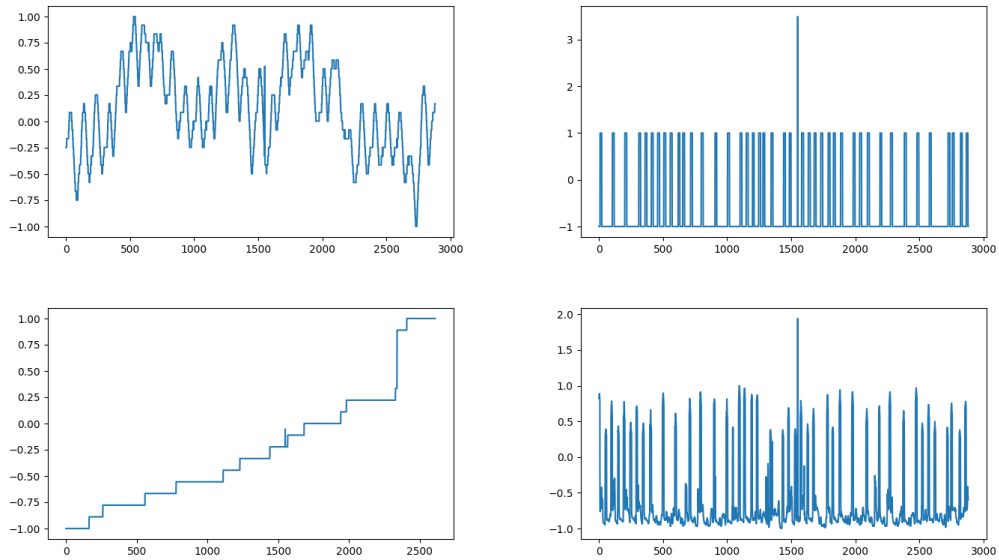


Figure 7: Visualization of po. In each of the time series a point outlier is injected at the 1550th value.

B.3 Time series Mixup

As described in section 3.3, inspired by Zhang et al. (2018) we use an adapted version of the MIXUP procedure as part of our framework.

We sample $\lambda \sim \text{Beta}(\alpha, \alpha)$ [13]. Using this λ we create a new training example as a convex combination of two examples from the batch:

$$x_{\text{new}} = \lambda x^{(i)} + (1-\lambda)x^{(j)}, \qquad \text{where } x^{(i)} \text{ and } x^{(j)} \text{ are two whole windows sampled from the batch}$$

$$y_{\text{new}} = \lambda y_s^{(i)} + (1-\lambda)y_s^{(j)}, \qquad \text{where } y_s^{(i)} \text{ and } y_s^{(j)} \text{ are the two corresponding labels.}$$

Note that, in addition to the new time series values x_{new}, the method also produces soft labels y_{new}, different to 0 or 1, which are used during training. [14]

Figure 8 shows example time series created using mixup. Each of the original time series is mixed up with its horizontal neighbor time series. We see that the newly created series have characteristics from both time series to create a new realistic time series. The patterns in (a) and (b) became a bit more noisy. The slope of (c) has the additional spiky pattern from (d) and the pattern in (d) now slowly ramps up.

C Further Results and Ablation Studies

C.1 Ablation Study on SMAP and MSL

Here we present a full ablation study on the SMAP and MSL datasets. We consider variations of the framework by removing some of its components, and train the model in each configurations twice. We report the average and standard deviation of these runs.

First, we observed that the contextual hypersphere formulation improves performance of the model. In the setting with all the data augmentation techniques "- contextual" the difference is not very big 1.98% F1 and 1.17% F1 on SMAP and MSL respectively. However, in the setting where none of the data augmentation is used, it makes a dramatic difference to use this formulation: 11.81% F1

[13] we set $\alpha = 0.05$, as this value gave the best generalization among the values that were tried in the experience of fig. 3b

[14] Our implementation of mixup can be found in file `src/ncad/model/mixup.py` of the supplementary code.

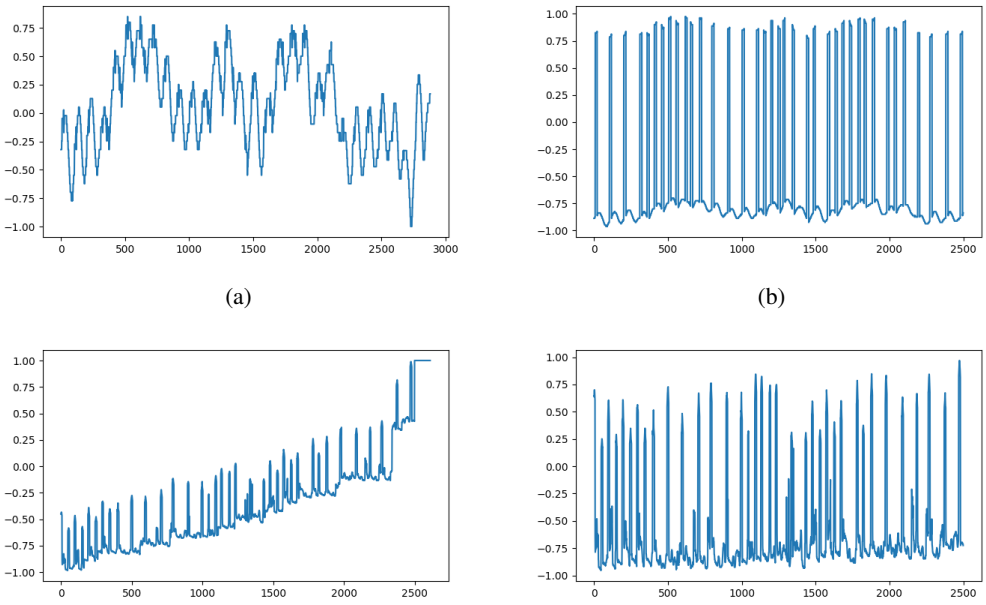


Figure 8: Visualization of time series mixup. Each of the series is "mixed up" with the its horizontal neighbor time series: (a) with (b) and (c) with (d)).

and 43.44% F1 on SMAP and MSL respectively. Further, we can see that solely with the contextual hypersphere and without relying on any data augmentation technique the model can achieve a very reasonable performance.

Table 3: F1 score of the model on SMAP and MSL

Model	SMAP	MSL
THOC (Shen et al., 2020)	95.18	93.67
NCAD w/ COE, po, mixup	94.45 ± 0.68	95.60 ± 0.68
- COE	88.59 ± 1.81	94.66 ± 0.22
- po	94.28 ± 0.45	94.73 ± 0.35
- mixup	92.69 ± 1.14	95.59 ± 0.01
- mixup - po	94.4 ± 0.43	94.12 ± 0.77
- mixup - COE	86.86 ± 0.7	91.7 ± 2.58
- COE - po	60.48 ± 9.7	42.02 ± 6.34
- mixup - COE - po	66.9 ± 2.01	79.47 ± 9.39
- contextual	92.47 ± 0.53	94.43 ± 0.15
- contextual - COE	91.86 ± 0.96	88.29 ± 0.43
- contextual - po	93.39 ± 0.61	90.68 ± 0.74
- contextual - mixup	94.37 ± 0.21	95.07 ± 0.14
- contextual - mixup - po	93.24 ± 0.31	90.89 ± 0.46
- contextual - mixup - COE	89.88 ± 2.53	87.26 ± 4.17
- contextual - COE - po	54.95 ± 2.62	32.05 ± 0.17
- contextual - COE - mixup - po	55.09 ± 1.0	36.03 ± 3.01

We also observed that both COE and po jointly improve the model performance. If we remove separately one of these elements, neither of the two tends to have a large impact on the performance. However, if none of them is used the performance drops drastically: by 43.98% F1 and 53.58% F1

for SMAP and MSL respectively, and the drop is even bigger when not using the contextual inductive bias.

It is interesting to note that it does not seem to be a good idea to use mixup as the only data augmentation method (at least in this unsupervised setting). In the setting where neither COE nor po are used, using mixup seems to significantly deteriorate the performance: by 6.42% and 37.45% on SMAP and MSL respectively. We conjecture that this is due to the fact that, for this datasets, there are no labels in the training data and so mixup does not allow to create soft-labels. In addition, mixup creates new time series that may not correspond to the original data distribution, these may deviate the learning away from the original data distribution.

C.2 Ablation study and Supervised benchmark on Yahoo dataset

Here we present the results of our method on the supervised Yahoo dataset. It is important to note that, since the only baseline method that we found evaluated their model with point-wise F1 score, this is also what we use here to make our results comparable.

Table 4: Supervised anomaly detection performance on YAHOO. Results for U-NET taken from (Gao et al., 2020).

	YAHOO		
MODEL	F1	PREC	REC
U-NET-RAW	40.3	47.3	35.1
U-NET-DE	62.1	65.1	59.4
U-NET-DEW	66.2	79.3	56.9
U-NET-DEWA	69.3	85.9	58.1
NCAD SUPERVISED	62.11	80.44	50.59
+ MIXUP	63.08	76.70	53.57
+ PO	**79.92**	74.96	85.57
+ COE	53.66	78.84	40.67
+ COE + MIXUP	59.85	78.89	48.21
+ PO + COE	58.36	54.89	62.30
+ PO + COE + MIXUP	67.32	88.38	54.36
- CONTEXTUAL	5.50	3.42	14.08
- CONTEXTUAL + PO	67.90	64.15	72.13
- CONTEXTUAL + COE	39.53	42.56	36.90
- CONTEXTUAL + PO + COE	55.25	43.87	74.60

The supervised approach proposed by Gao et al. (2020) is based on a U-net architecture, which is combined with preprocessing (using robust time series decomposition), loss weighting (to up-weight the rare anomalous class), and several forms of tailored data augmentation applied to the time series (keeping the labels unchanged). They report results for four variants: U-NET-RAW (plain supervised U-net on raw data), U-NET-DE (applied to residual after preprocessing), U-NET-DEW (with loss weighting), U-NET-DEWA (with loss weighting and data augmentation).

Using only the true labels but no data augmentation (NCAD SUPERVISED), our approach significantly outperforms U-NET-RAW, and performs on-par with U-NET-DE, without relying on time series decomposition and using an arguably much simpler architecture.

When we use the po data augmentation, our approach outperforms the full U-NET-DEWA by a large margin, hinting at the possibility that addressing the class imbalance problem by creating artificial anomalies is more effective than using their strategy of loss weighting while keeping the labels intact.

In the supervised setting, injecting the generic COE anomalies (either individually or in combination with po) hurts performance, presumably by steering the model away from the specific kind of anomalies that are labeled as anomalous in this data set. On the other hand, adding MIXUP generally improves performance. The contrastive loss is crucial for good performance, as shown by the rows labeled - CONTRASTIVE, where it is replaced with a standard softmax classifier.

D Model Implementation and Training

At the core of the NCAD framework, we use a *single* Encoder, $\phi(\cdot;\theta)$, to produce time series representations. The same encoder is applicable to all the time series in a given dataset, and it is used to encode both the full windows and the context windows. The parameters of the encoder, θ, are learned via Mini-Batch Gradient Descent, aimed at minimizing the classification loss discussed in appendix A.3.

Training mini-batches of size B are created by first randomly selecting b_s series from the training set, and then taking b_c random fixed-size windows from each [15]. Data augmentation strategies described in section 3.3 are applied to these windows, creating additional examples which are incorporated to the batch. The number of augmented examples is controlled as a proportion of the original batch, using two additional hyperparameters: r_{coe} and r_{mixup} for COE and Mixup, respectively. The size of the training batch is therefore

$$B = b_s b_c + \lfloor b_s b_c r_{coe} \rfloor + \lfloor b_s b_c r_{mixup} \rfloor$$

Our implementation[16] is based on PyTorch (Paszke et al., 2019) and PyTorch Lightning (Falcon, 2019). We used the default initialization defined in PyTorch, and the *YOGI* optimizer (Zaheer et al., 2018) for all our experiments. We use standard AWS EC2 ml.p3.2xlarge instances with a single-core Tesla V100 GPU. Training and hyperparameter tuning was aided by AWS Sagemaker (Liberty et al., 2020), training takes on average 90 minutes for each dataset.

D.1 Model Hyperparameters

Hyperparameters in our framework con be mainly divided in four categories:

1. **Encoder architecture**: Number of TCN layers, TCN kernel size, embedding dimension, embedding normalization.
2. **Data augmentation**: r_{coe}, r_{mixup} (described above).
3. **Optimizer**: learning rate, number of epochs.
4. **Mini-batch cropping**: window length, suspect window length, b_s, b_c.

For YAHOO, KPI and SWaT, validation labels are available, so we use a Bayesian optimization (Perrone et al., 2020) for hyperparameter tuning, maximizing the F1 score on the validation set. We restricted the search of hyperparameters to only "sensible" values for most of the hyperparameters (e.g. max. 10 TCN layers, max. 256 dimensions for the embedding, max. 2.0 for the augmentation rates, etc.). Lengths of the window and suspect window are set by observing the lengths and seasonal patterns of the training dataset, so that it covers at least one cycle and this seasonality could be encoded in the representation. We use early stopping and keep the model with the lower validation F1, which is then evaluated on the test dataset and the result is reported.

For SMAP, MSL and SMD, we do not have validation data to pick the hyperparametes, so we use default values that seemed to work well for the other datasets. It is not possible to do early stopping either, so we keep the model resulting of training until the last epoch, which is then evaluated on the test dataset and the result is reported.

In all cases, we align our experimental protocol with prior works and report $F1$ scores computed by choosing the best threshold on the test set.

We provide hyperparameter configuration files in the supplementary code, which allow to replicate our benchmark results in section section 4.

[15] `num_series_in_train_batch` and `num_crops_per_series` in the supplementary code

[16] open-source code available at `Anonymous Github repository`

E Datasets and External Assets

We use the following datasets to compare the performance of NCAD to other methods:

Soil Moisture Active Passive satellite (SMAP) and **Mars Science Laboratory rover (MSL)** Hundman et al. (2018), the datasets are under the custom license `https://github.com/khundman/telemanom/blob/master/LICENSE.txt`.

Secure Water Treatment (SWaT) (Mathur & Tippenhauer, 2016) This dataset is distributed by the ITrust Centre for Research in Cyber Security `https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/`, we were not able to find the precise license of the dataset.

Server Machine Dataset (SMD) (Su et al., 2019) is distributed under the MIT License `https://github.com/NetManAIOps/OmniAnomaly/blob/master/LICENSE`.

YAHOO A dataset published by Yahoo labs, `https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70`, we were not able to find the precise license of the datase beyond the ReadMe.txt specifying that the dataset could be used for non-commercial research purposes.

KPI (kpi) we were not able to find the precise license of the dataset.

Additional assets In addition to the datasets, we use existing code for the TCN encoder from `https://github.com/White-Link/UnsupervisedScalableRepresentationLearningTimeSeries` which is under the Apache License Version 2.0. We also use the evaluation code from `https://github.com/NetManAIOps/OmniAnomaly/blob/master/donut_anomaly/eval_methods.py` which is under the MIT License. We use the standard Python library numpy Harris et al. (2020), which is under the BSD 3-Clause "New" or "Revised" License `https://github.com/numpy/numpy/blob/main/LICENSE.txt`.

We make our code available, licensed under the Apache License, Version 2.0.

All the dataset and code that we use in this work is openly available under licences that allow to use them, as a result we did not seek additional consent from their creators. None of the datasets contains personally identifiable information, nor do they contain offensive content.