2022-12-05

The interaction of normalisation and clustering in sub-domain definition for multi-source transfer learning based time series anomaly detection

Matthew Nicholson
ADAPT Centre, Trinity College Dublin, matthew.nicholson@adaptcentre.ie

Rahul Agrahari
ADAPT Centre, Trinity College Dublin, rahul.agrahari@adaptcentre.ie

Clare Conran
ADAPT Centre, Dublin City University, clare.conran@adaptcentre.ie

See next page for additional authors

Recommended Citation

Matthew Nicholson, Rahul Agrahari, Clare Conran, Haythem Assem, John D. Kelleher, The interaction of normalisation and clustering in sub-domain definition for multi-source transfer learning based time series anomaly detection, Knowledge-Based Systems, Volume 257, 2022, 109894, ISSN 0950-7051, DOI: 10.1016/j.knosys.2022.109894.

Funder: SFI ADAPT Centre

Knowledge-Based Systems 257 (2022) 109894

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

The interaction of normalisation and clustering in sub-domain definition for multi-source transfer learning based time series anomaly detection

Matthew Nicholson [a,1], Rahul Agrahari [a,1], Clare Conran [b,1], Haythem Assem [c,2], John D. Kelleher [d,*,2]

[a] ADAPT Research Centre, Trinity College Dublin, Ireland
[b] ADAPT Research Centre, Dublin City University, Ireland
[c] Huawei Research, Ireland
[d] ADAPT Research Centre, Technological University Dublin, Ireland

ARTICLE INFO

Article history:
Received 16 March 2022
Received in revised form 11 September 2022
Accepted 12 September 2022
Available online 17 September 2022

Keywords:
Anomaly detection
Transfer learning
Time series analysis
Cloud infrastructure

ABSTRACT

This paper examines how data normalisation and clustering interact in the definition of sub-domains within multi-source transfer learning systems for time series anomaly detection. The paper introduces a distinction between (i) clustering as a primary/direct method for anomaly detection, and (ii) clustering as a method for identifying sub-domains within the source or target datasets. Reporting the results of three sets of experiments, we find that normalisation after feature extraction and before clustering results in the best performance for anomaly detection. Interestingly, we find that in the multi-source transfer learning scenario clustering on the target dataset and identifying subdomains in the target data can result in improved model performance, as compared to identifying sub-domains through defining clusters using the multi-source dataset.

1. Introduction

Cloud infrastructures are now a standard platform for deploying software. Consequently, the reliability of these systems is critical to the success of many technology companies. This has led to a growth in research focused on identifying anomalous behaviour in cloud systems. Cloud system monitoring data is naturally time series based, and so identifying anomalous behaviour in these systems is often framed in terms of time series anomaly detection.

One of the major challenges in anomaly detection is the imbalance in the data: anomalies by definition are rare events. Therefore, applying supervised machine learning methods to anomaly detection can require a substantial annotation effort. For cloud services, however, the dynamic nature of the service usage and deployment, as well as the variety of components monitored makes the annotation of anomalies very expensive. At the same time, unsupervised methods are unlikely to produce the high level of accuracy required for the effective and timely maintenance of these systems [1]. Transfer learning has the potential to produce the high-accuracy of supervised models while at the same time reducing the annotation effort required to develop these systems. Transfer learning involves using information learned to carry out a task in one domain to aid learning in another domain, and for transfer learning to work well the two domains should be similar [2].

Within cloud services the optimal definition of domains is non-trivial. For example, for a given cloud service there is likely a stream of time series monitoring data related to CPU usage and also a stream of monitoring data related to GPU usage. In such a scenario it may be optimal to consider CPU and GPU usage as distinct domains. However, it may also be useful to consider the combination of CPU and GPU usage as a single domain. The reason for this is that within a particular cloud system the configuration and usage of CPUs and GPUs may be such that the data streams generated by both of these processes are similar enough that they can be usefully modelled as being sampled from a single distribution, in which case merging these two streams of data will result in more labelled data being available for anomaly detection across both these services. Furthermore, across different cloud services, the CPU usage from one service may be more similar to the GPU usage from another service as compared with the CPU usage of this other service.

* Corresponding author.
E-mail address: john.d.kelleher@tudublin.ie (J.D. Kelleher).
1 These authors contributed equally to this work, and are joint first authors.
2 These authors jointly supervised the work, and are joint senior authors on this work.

This paper examines how the definition of domains affects the performance of anomaly detection within cloud services. More specifically it explores how normalisation and clustering processes interact in the definition of domains and the downstream effect of these interactions on the accuracy of anomaly detection systems in cross-domain time series anomaly detection. The paper is structured as follows: we begin by introducing the research challenge of multi-source transfer learning, we then distinguish between the use of clustering as a mechanism for anomaly detection versus sub-domain definition, next we describe the different methods of normalisation that we assess, this is followed by an experiments section that reports 3 different experiments, the paper finishes with a conclusions section which overviews the main findings of the research.

2. Multi-source transfer learning & related work

Following the standard definition of transfer learning – as set out in e.g. [3,4] – a domain $\mathcal{D}$ is composed of two parts:

1. a feature space $\mathcal{X}$ that defines all possible feature vectors, and
2. a marginal distribution $P(X)$ over the data sample X, where $X = \{x_1, \cdots, x_n\} \in \mathcal{X}$ (i.e., x_i is the ith instance in X).

Similarly, a task $\mathcal{T}$ in a domain $\mathcal{D}$ is also composed of two parts:

1. a label space $\mathcal{Y}$ (for anomaly detection $\mathcal{Y} = \{true, false\}$), and
2. a predictive function $f(*)$ which is not observable but which can be learned from a dataset of instances and label pairs $\{x_i, y_i\}$.

Given these definitions of a domain $\mathcal{D}$ and a task $\mathcal{T}$, transfer learning is defined as using information learned from the source domain $\mathcal{D}_S$ to do task $\mathcal{T}_S$ to improve the learning in the target domain $\mathcal{D}_T$ of the target predictive function $f_T(*)$ for task $\mathcal{T}_T$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$.

The transfer learning constraint that $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$ means that transfer learning can be used when the domains are different (either because $\mathcal{X}_S \neq \mathcal{X}_T$ or $P(X_S) \neq P(X_T)$ or when the tasks are different (because $\mathcal{Y}_S \neq \mathcal{Y}_T$ or $f_S(*) \neq f_T(*)$). Corresponding with the different ways that domains can differ Pang & Yang [3] identify four transfer learning methods based on what is being transferred:

1. instance-based methods attempt to correct the differences in the marginal distribution (i.e., these methods attempt to make $P(X_S)$ and $P(X_T)$ more similar) by weighting the samples in the source domain based on their similarity with the samples in the target domain,
2. feature-representation transfer methods which focus on using the data sample from the source domain X_S to learn a useful feature representation $\mathcal{X}$ for the target domain (these methods often use dimensionality reduction or unsupervised/semi-supervised methods to attempt to usefully align $\mathcal{X}_S$ and $\mathcal{X}_T$ with the knowledge transferred between the domains being the learned feature representation),
3. parameter transfer methods focus on sharing parameters between $f_S(*)$ and $f_T(*)$ (in other words, they assume that the conditional probability distribution for the task in the source and target domain are similar),
4. relation knowledge transfer is applicable in scenarios where within each domain (source and target) relationships exist between entities in the domain and the relationships among the data in the source and target domains are similar and so the distribution of relationships within the source domain can help on the task in the target domain.

In time series anomaly detection it is standard practice to convert each data point in a time series into a fixed-width feature vector before applying an anomaly detection decision process to the data point. This is done by applying a feature extraction process to the raw time series. Fig. 1 illustrates this feature extraction process.

The same feature extraction process can be applied to any univariate time series and consequently the feature spaces are the same across the datasets we consider (i.e., $\mathcal{X}_S = \mathcal{X}_T$). Technically, when $\mathcal{X}_S = \mathcal{X}_T$ the transfer learning process is categorised as homogeneous transfer learning [4]. Furthermore, all the datasets we consider are focused on anomaly detection and so $\mathcal{Y}_S = \mathcal{Y}_T$ (anomaly = true or false). However, the source and target domains may differ in terms of the marginal distribution $P(X_S) \neq P(X_T)$ (i.e., the task involves domain adaptation) and also in terms of the conditional probability distributions for the task $f_S(*) \neq f_T(*)$. Consequently, for our scenarios, instance-based transfer learning is most suitable and it is the approach we will focus on in this work.

One of the challenges in applying instance-based transfer learning to cross-domain cloud service anomaly detection is that both the source and target domain datasets are composed of multiple heterogeneous data streams. Consequently, not every instance in the source domain is relevant to a particular target data point. Transferring irrelevant or incorrect data from the source to the target domain both increases computational overhead and can also result in reduced learning performance in the target domain, a phenomenon which is known as *negative transfer* [5,6]. As a result, it is not necessarily a good idea to use all the source data to pre-train the task model [6].

One approach to addressing the problem of multiple distinct data sources (or distributions) within the source domain is to train a separate model for each data source/distribution and then combine these separate models into a single model. Adopting this perspective, Christodoulidis et al. [7] is an example of research on transfer learning that directly addresses the problem of multi-source transfer learning: i.e., the source dataset contains data from different sources. The focus of Christodoulidis et al. [7] is to develop a computer-aided diagnosis system for lung pattern analysis. Due to the difficulty of sourcing medical images and the cost of annotation associated with medical data they adopt a transfer learning approach. The diagnosis of interstitial lung disease from CT scans involves recognising textural patterns. Consequently, they created a source domain dataset by combining 6 general texture databases. The proposed approach to transfer learning involved: (a) training a separate CNN for each of the 6 general texture databases, and (b) creating multiple ensembles of the 6 CNNs resulting from the first phrase by repeatedly using an iterative forward-selection algorithm to select different subsets of the CNNs for each ensemble, (c) creating a single ensemble that merges the ensembles created in the previous step by averaging their outputs, and (d) using knowledge distillation (also known as model compression) to compress the huge ensemble into a smaller form: more specifically a final model is trained using the soft targets (i.e., the class probabilities) generated for each training instance by the large single ensemble. Yi et al. [8] is an example of more recent work that tackled the challenge of transfer learning in the context where the source domain contains data sampled from multiple distributions. Xu et al. distinguish between the problems of multi-source and multi-component transfer learning: in multi-source transfer learning the multiple sources of data in the source dataset are caused by known factors such as distinct data collection techniques or distinct data sources, whereas the causes of differences between distinct components in the source dataset are not usually known in advances. One consequence of this distinction between multi-source and multi-component is that in a multi-source scenario

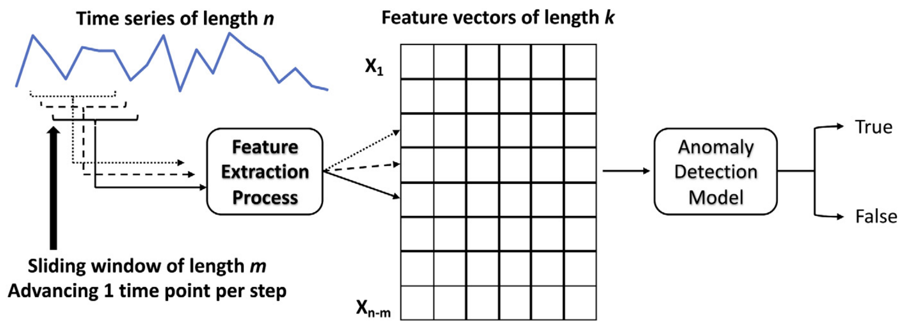
Time series of length *n*
Feature vectors of length *k*
X_1
X_{n-m}
Feature Extraction Process
Anomaly Detection Model
True
False
Sliding window of length *m*
Advancing 1 time point per step

Fig. 1. Feature extraction from univariate time series data resulting in each raw time series data point being represented by a vector of features.

it is possible to use prior knowledge to cleanly and explicitly disentangle the distinct data distributions in the source domain (e.g., Christodoulidis et al. begin their approach by separating each of the 6 general textual datasets and training a separate model for each of these) whereas in a multi-component scenario the different data distributions in the source domain must be identified and disentangled using some form of data analysis. Yi et al. [8] propose the use of distance-based cluster methods (e.g., k-means, PCA, spectral clustering) to identify and extract separate components (i.e., clusters) from the source domain. They then use a sample of labelled examples from the target domain to train a separate Mahalanobis distance-based transfer metric for each component. Each of these transfer metrics can be used in conjunction with the data points within the corresponding component to generate predictions for the sample of labelled target domain samples by using it as the distance metric for a kNN model. An ensemble model can then be created that averages over the predictions made by each component's kNN model. The training of a transfer metric for a component involves learning weights for each of the data points within the component (cluster) the transfer metric is associated with and adjusting the covariance matrix of the components Mahalanobis distance metric to reduce the prediction error on the labelled target domain samples. Moon & Carbonell [9] is another example of research that use pre-clustering on the source domain to avoid negative transfer in transfer learning. Similar to Yi et al. [8] they ultimately create a single model that weights (in this case using an attention mechanism within a deep network) the information from each of the separate clusters of source data to make predictions in the target domain.

Zhang et al. [10] is a recent example of work that addresses the multi-source challenge within transfer learning in the context of time series anomaly detection in cloud services. As a result, it is potentially the most directly relevant paper to the current work. Similar to the Moon & Carbonell [9], and Yi et al. [8] they apply clustering methods (specifically k-means) to the source dataset to identify distinct subsets of samples. They use the concept of a sub-source domain to refer to identified clusters: a sub-source domain in Zhang et al. [10] has a similar meaning to a component in Yi et al. [8]. Indeed, they motivate the definition and use of sub-source domains in transfer learning as follows:

In transfer learning, we should guarantee that the source and target domain come from similar fields (such as similar monitoring data) or own similar characteristics (such as trend or period) [10, pg. 6]

Once these sub-source domains are identified the labelled data within each sub-source domain is used to train an anomaly detection model for that sub-source domain. Where Zhang et al. [10] is distinctive relative to previous work in multi-source transfer learning is that they do not merge the sub-source domain classifiers into a single model. Instead, each target domain data point is assigned to a single sub-source domain based on its proximity in the feature space to the sub-source centroids and is then processed by the anomaly detection model trained using the sub-source domain data.

All of the research reviewed in this section highlights the importance of disaggregating multi-source datasets to reduce negative transfer. In Christodoulidis et al. [7] this disaggregation was achieved by separating the source dataset based on the original datasets that were combined to create it. Whereas, in the other examples of research [8–10] clustering was used to identify component/sub-source domains. However, clustering processes can identify sub-domains based on many different perspectives: normal versus anomalous points, distinctions between types of anomalies (e.g., point, contextual, or collective [11]), data points with similar values, and so on. Consequently, the type of sub-domains identified through a clustering process may have a significant impact on the downstream performance of the anomaly detection system. In the next section, we will review the use of clustering in time series anomaly detection and distinguish between its use as a direct method for anomaly detection versus its use for the identification and definition of sub-domains.

3. Clustering as anomaly detector versus sub-domain identification

Within the literature, on anomaly detection, there is a tradition of research that use clustering techniques to perform anomaly detection. Ahmed et al. [12] identify three assumptions that are generally made when clustering techniques are used to directly carry out anomaly detection:

1. It is possible to create clusters of only normal data and so new anomalous data points can be identified as points that do not fit existing clusters
2. If clusters contain both normal and anomalous points then normal points are more centrally located within a cluster relative to the anomalies and so anomalies can be identified based on the distance to the cluster centroid
3. Where the clustering algorithm generates clusters of different sizes than smaller clusters are considered to primarily contain anomalies and so points belonging to these small clusters are considered as anomalies

For example, Münz et al. [13] used k-means clustering for anomaly detection in network traffic data. Specifically, k-means clustering was used to segment unlabelled data into distinct clusters with the assumption that normal and anomalous data will naturally form different clusters. The resulting clusters are then categorised as normal or anomalous clusters using a combination of heuristic rules and manual eyeballing. For their experiments Münz et al. set k = 2 and so they created one normal and one anomalous cluster. Once the clusters have been created and labelled as normal or anomalous new data points are then processed and identified as anomalous if: (a) the distance between the point and the centroid of the normal cluster was greater than a pre-specified threshold, or (b) a point was closer to the centroid of the anomalous cluster than the normal cluster. This combination of rules covered the possibility that a point could be inside the distance threshold to the normal cluster but still be closer to the centroid of the anomalous cluster than the normal cluster. Münz et al. separated their data into different predefined service classes (e.g. TCP, UDP, or ICMP traffic) and ran a separate clustering process for each service class. Splitting the data in this way can be understood as defining separate sub-domains within which anomaly detection using clustering is performed.

More recently, Syarif et al. [14] carried out a benchmarking of five different clustering algorithms (k-Means, improved k-Means, k-Medoids, EM clustering, and distance-based outlier detection) for network anomaly detection. Their results indicated that the distance-based outlier detection algorithm had the best performance with an accuracy of 80.15%, however, all the algorithms had a high false positive rate ($> 20\%$). Syarif et al. used the NSL-KDD intrusion dataset for their experiments which was made available as part of the KDD Cup 1999. This dataset contains TCP data from a single local-area network, so it can be considered to be from a single domain.

In contrast with the above work, in a cloud service setting time series data streams are generated from multiple different components [10]. What is more, cloud architectures can be dynamically configured with new components added to the architecture. In these contexts, cloud services data sets can be considered multi-source and so clustering can be applied to identify and disentangle different sub-domains/components within the data. However, as the work in Münz et al. [13] and Syarif et al. [14] demonstrates clustering can result in distinctions between normal and anomalous points being identified, rather than identifying sub-domains/components based on underlying data distributions.

Fig. 2 illustrates a multi-source time series anomaly detection scenario where clustering results in normal and anomalous clusters. In this figure the source dataset is composed of data streams from 4 different data generation processes: P1, P2, P3 and P4. P1 samples data from a normal distribution, P2 from a cyclical distribution, P3 from a uniform distribution, and P4 from a normal distribution. Each of these processes contributes five data points to the multi-source dataset, within the figure the source of each data point is visually encoded using different shapes and the anomalies within each data stream are marked using **X**. The clustering process defines 3 clusters (i.e., subdomains) one containing all the anomalies and two others that contain mixtures of non-anomalous data points from different sources. Fig. 3 illustrates the same multi-source time series anomaly detection scenario as that shown in Fig. 2. However, in this instance, the clustering process has identified that the data points generated from processes P1 and P4 are similar in that they are both sampled from a similar normal distribution and so the clustering process had defined a sub-domain that combines the data points from these two data sources.

Figs. 2 and 3 can be understood as illustrating the ends of a spectrum of possible outcomes in terms of clustering-based sub-domain definition. Many clustering processes will result in some blend of these two possible results. However, one open research question related to these alternative outcomes is how the pre-processing of the data before clustering affects the resulting sub-domain definition (if at all)? For example, as we will discuss in the following section, we may choose to normalise the data in different ways before clustering and some of these normalisation processes may make data from different data sources appear more similar, and/or make anomalous data points more distinct. If this is the case then the choice of normalisation process may nudge a clustering process to produce sub-domains definitions that are more similar to Fig. 2 or Fig. 3. Another very related question is whether either of the outcomes illustrated in Figs. 2 and 3 result in better anomaly detection performance, both within a domain and in the transfer learning context. In the next section, we introduce the different normalisation options we consider and following that we present the experiments we have run to answer these questions.

4. Feature normalisation and clustering methods

Fig. 1 illustrates a standard data handling process for time series data where a feature extraction process is applied to the raw time series data resulting in each data point in a time series being represented by a vector of features. The feature engineering process helps us to identify the underlying information from the time series which sometimes is not obvious from the raw data, such as the underlying temporal and spatial trends within the locality of the data point within the time series. The features are extracted from fixed-length moving windows which help to preserve the temporal information embedded in the data and the processing over it to create feature vectors enhances the hidden information such as moving average, variance, trend, seasonality etc. Previously there have been multiple works on extracting useful information from the time series for performing multiple tasks, see for example [15,16]. Agrahari et al. [17] report a set of experiments that assessed different feature representations for instanced-based cross-domain anomaly detection in cloud services based on univariate time series data. These experiments indicate that the Catch22 feature set proposed by [16] augmented with two extra features (i.e., moving average and moving variance) had the best performance. Consequently, we use the extended feature set in this work and refer to it as the Catch24 feature set.

In this paper, however, we are focusing on normalisation at different stages of feature engineering to understand how normalisation impacts the identification of sub-domains via clustering and also on anomaly detection performance. The form of normalisation we consider is standardisation. In standardisation, the values are centred around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the final distribution has a unit standard deviation. For this work, a key advantage of standardisation, as compared with for example range normalisation, is that standardisation does not have a bounding range and so outliers within the data are not affected by the process of standardisation. Eq. (1) defines the standardisation process where x' is the standardised value, x is the original value, μ is the mean and σ the standard deviation of the original values

$$x' = \frac{x - \mu}{\sigma} \tag{1}$$

In the feature extraction process shown in Fig. 1, there are two places in this process where standardisation can be applied before clustering these are:

Before feature extraction each time series in a dataset is standardised to have zero mean and unit standard deviation before the feature extraction process. It is important to note that each of the datasets we use in our experiments is composed of multiple files with each file containing a separate time series of univariate cloud-service monitoring data. Each of these files contains a time series from a single source in the cloud service, e.g. a file might contain a times-series sampled from GPU usage monitoring data or CPU usage monitoring data but not from both. Also, within each dataset, there are multiple files from each source. This structure of the data means that standardising

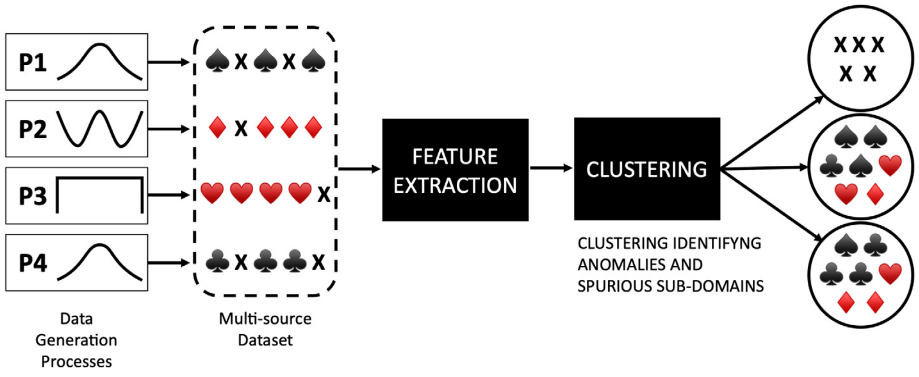
P1
P2
P3
P4
Data Generation Processes
Multi-source Dataset
FEATURE EXTRACTION
CLUSTERING
CLUSTERING IDENTIFYNG ANOMALIES AND SPURIOUS SUB-DOMAINS

Fig. 2. Clustering identifying anomalies and 'spurious' subdomains.

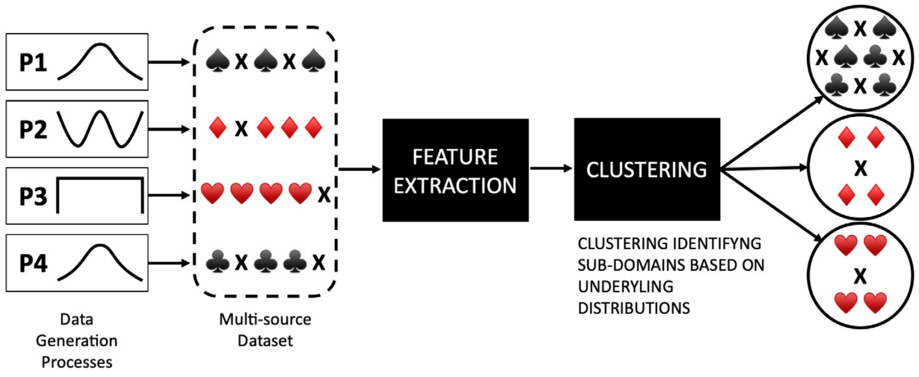
P1
P2
P3
P4
Data Generation Processes
Multi-source Dataset
FEATURE EXTRACTION
CLUSTERING
CLUSTERING IDENTIFYNG SUB-DOMAINS BASED ON UNDERLYING DISTRIBUTIONS

Fig. 3. Clustering identifying subdomains based on underlying data distributions.

before feature-extraction results in the data in each file being standardised relative to the mean and standard deviation of the data in that file. This form of standardisation may result in data from different files (and hence from different monitoring services) appearing more similar and hence being clustered together during subsequent clustering. Also, depending on the relative contribution of anomalous values within a file to the calculation of the mean and standard deviation for the file, standardising before feature extraction at the file/time series level may result in the normal data within a file being pushed closer together and the anomalous points becoming more salient within the file.

After feature extraction when normalisation is applied after feature extraction the standardisation process is applied to each feature column. This means that a separate mean and standard deviation are calculated for each of the 24 features in our feature set and the set of values for each feature across the entire dataset are standardised to have a mean of zero and a standard deviation of one. Note that in this scenario the set of values used to calculate a mean and/or standard deviation are sampled across multiple files/time series and so from data points sampled from multiple data generation processes within the cloud-service architecture. As a result, the differences in ranges of values from different data sources may be indirectly emphasised, with feature values calculated from time series with similar ranges appearing more similar.

A third, baseline, alternative is to do **no normalisation** before clustering/sub-domain identification. The following sections report a set of experiments that explore how these different approaches to normalisation (none, before feature extraction, and after feature extraction) affect the structure of the sub-domains created by clustering and the performance of anomaly detection systems.

5. Experimental design

We explore the interaction between data normalisation, clustering, subdomain analysis and downstream anomaly detection performance through three experiments. The first experiment examines which normalisation method results in the best performance on anomaly detection and whether there is a statistical difference in performance between different normalisation methods. The second experiment analyses whether differences exist in the structure of the subdomains created by the clustering when different forms of normalisation are applied to the data before clustering. The structure of the domains is analysed in terms of the distribution of anomalies across clusters and also the distribution of data points from different time series across clusters. The third experiment focuses on finding the best performance on cross dataset anomaly detection when doing transfer learning when the source dataset is created by merging multiple datasets, again checking whether there is a statistical difference in performance between different normalisation methods and looking into the impact of the number of clusters.

We use six datasets in our experiments: NAB (AWS and Twitter) [18], Yahoo (Real and Artificial) [19], IOPS KPI,[3] and Huawei.[4] Each dataset contains multiple files and each file contains one time series. Table 1 provides summary statistics for each dataset. Furthermore, following the results reported in [17] we use the Catch24 feature set in combination with random forest models for all our experiments. As illustrated in Fig. 1 the conversion of a time series into the Catch24 feature set involves passing a sliding window along the time series using a step size of 1, and

[3] Available from: https://github.com/NetManAIOps/KPI-Anomaly-Detection.

[4] From the 2020 Huawei anomaly detection competition: https://huawei-euchallenge.bemyapp.com/ireland.

Table 1
Summary statistics for the datasets used in the experiments.

Dataset	# of points	% of anomalies	# of time series	Mean length
Yahoo real	91k	1.76%	64	1415
Yahoo artificial	140K	1.76%	100	1415
IOPS	3M	2.26%	29	105985
AWS	67K	4.57%	17	67740
Twitter	142K	0.15%	10	142765
Huawei	54K	4.19%	6	9056

each time the window is moved one time-step forward a Catch24 feature vector is generated by calculating the 24 features using the segment of the time series covered by the window. Each feature vector represents the right-most point in the time series segment covered by the window, and so the feature vector is labelled with the same label as this right-most point (i.e., anomaly = true or false). In our experiments, the width of the sliding window was set separately for each time series in a dataset to the periodicity of the time series, as calculated by the AUTOPERIOD method [20].

In all three experiments, we use k-means as our clustering method and k-means++ to initialise the centres of k-means clustering [21]. Our motivation for using k-means is twofold. First, k-means is the most popular form of clustering method within the previous literature on anomaly detection and, more specifically, multi-source transfer learning anomaly detection. For example, [8–10,13] all use k-means as a clustering method in their work. Second, k-means time complexity $O(nKI)$ and space complexity $O(n(D+K))$ where K is the number of centres, D the number of dimensions and I the number of iterations to converge) makes the use of k-means within our experiments appropriate. We did also consider using other forms of clustering methods in our experiments. However, hierarchical clustering methods are not suitable for our task due to the need to calculate and recalculate full pairwise distance matrices with these methods which results in a processing time complexity of $O(n^3)$ and space complexity of $O(n^2)$. Density-based clustering methods such as DBSCAN have a better complexity profile (worst case time complexity is $O(n^2)$ but in low dimensional data where spatial indexes work well the average run-time complexity is $O(n\ log\ n)$ and space $complexityisO(n)$) as compared with hierarchical methods. However, density-based clustering methods use a global density when identifying clusters, consequently, variation in densities across a dataset can cause these methods to merge dense clusters or miss sparse clusters [22]. Our experiments with fitting DBSCAN to our datasets suggest that density methods may not be appropriate for multi-source datasets due to the variation in density across the datasets.[5]

Finally, Wu & Keogh [23] recently highlighted the problem of noise in the labels of several time series anomaly detection benchmark datasets. To mitigate the effect of this noise on our results, in our experiments, we use cross-validation and report the mean of each metric as well as the 95% confidence interval around this mean.

6. Experiment 1: Which normalisation method gives the best performance? (within dataset baseline scenario)

The main research questions explored in this experiment were as follows:

1. Which of the three normalisation options (no normalisation, normalising before feature extraction, and normalising after feature extraction) results in the best performance in anomaly detection?
2. Is there a statistical difference between performances based on different normalisation methods?

For this experiment, each of the six datasets was kept separate and the results from the experiments on each of these datasets are then averaged. As a result, there was no transfer learning between the different datasets in this experiment. Algorithm 1 lists the process followed in this experiment. Given the 6 datasets, clustering is done for 9 values of k (2–10) and 5-fold cross-validation is carried out for each clustering. Then the average precision, recall and F1 for each form of normalisation are calculated across 6 * 9 * 5 = 270 values.

Algorithm 1 Experiment 1 Methodology.

```
for each normalisation type: none, before feat. ext., after feat. ext. do
    for each dataset do
        for k in 2 through 10 do
            Perform clustering with k-means++ on the dataset
            for each fold in a 5-fold cross validation do
                Train an anomaly detection model for each cluster
                using the training points belonging to that cluster
                For each cluster's anomaly detection model
                calculate precision, recall and F1 using the tests
                points belonging to that cluster
Calculate an average precision, recall and F1 for each form
of normalisation across all folds, and all values of k and all
datasets.
```

The results in Table 2 indicate that normalising after feature extraction results in the best performance in terms of precision, recall and F1. The improvement in performance achieved with normalisation after feature extraction as compared with the other forms of normalisation is not statistically significant in terms of recall, but it is statistically significant for precision and F1.

7. Experiment 2: Understanding the relationship between subdomain structures and normalisation strategies

In this experiment, we analyse how the structure of the subdomains that are created by a clustering process changes as the

[5] In more detail, DBSCAN specifies density through the interaction of two hyper-parameters: (1) epsilon specifies a radius around a point, and (2) MinPts specifies the minimum number of points that must be within epsilon of a point for that point to be considered a core point. Points that are identified as either a core point or as a point within epsilon of a core point are included in the clusters identified by the algorithm and all other points are categorised as noise. We ran a grid search over these two hyper-parameters and observed that when MinPts was set to the minimum recommended value of 3 the algorithm returned over 2500 clusters with around 5% of the data points considered as noise (i.e., not included in any cluster), and importantly over 5% of the points labelled as anomalies in the data were in this noise category. Furthermore, as the MinPts parameter increases the number of clusters identified by the algorithm rapidly decreases and the number of points considered to be noise also grew rapidly with the percentage of noise points that are anomalies growing even more rapidly (e.g. when MinPts= 8 DBSCAN identified ≈1100 clusters and categorised over 40% of the data as noise including over 15% of the anomalies in the dataset. Due to both the large variation in the number of clusters identified as the hyper-parameters were varied and the disproportionate amount of anomalies categorised as noise by the algorithm we consider that density-based methods were not suitable for our work.

Table 2
Average precision, recall and F1 for each type of normalisation across all 6 datasets. Calculated to 95% confidence interval. Population sizes 270.

Data	Average precision	Confidence (Error margin)	Range
After feature extraction	0.9891	±0.0010	(0.9881, 0.9901)
Before feature extraction	0.9846	±0.0018	(0.9828, 0.9864)
No normalisation	0.9847	±0.0018	(0.9829, 0.9865)
Data	Average recall	Confidence (Error margin)	Range
After feature extraction	0.9153	±0.0050	(0.9103, 0.9204)
Before feature extraction	0.8936	±0.0049	(0.8887, 0.8985)
No normalisation	0.9042	±0.0062	(0.8979, 0.9104)
Data	Average F1	Confidence (Error margin)	Range
After feature extraction	0.9504	±0.0030	(0.9474, 0.9534)
Before feature extraction	0.9365	±0.0030	(0.9334, 0.9395)
No normalisation	0.9420	±0.0038	(0.9382, 0.9458)

normalisation applied to the data before clustering is changed. Figs. 2 and 3 illustrate the two extremes of the spectrum of subdomain definitions that we expect: at one extreme (Fig. 2) the clustering process distinguishes between anomalies and non-anomalies, and at the other extreme (Fig. 3) the clustering process is defining sub-domains in terms of similarities between the underlying distributions that the data points are sampled from. Each of these clustering outcomes can be characterised and contrasted in terms of the distribution of anomalies across clusters and the distribution of points from files across clusters. For example, in Fig. 2 the anomalous points are all grouped in the same cluster, but the points from individual data generation processes are distributed across the clusters (e.g., if we include the P1 anomalous points, the points generated from P1 are distributed across all three of the clusters). By contrast, in Fig. 3 the anomalous points in the dataset are distributed across all three clusters and the points from each data generation process end up in the same cluster.

For this experiment, we used Shannon's measure of entropy as a measure of distribution across clusters. Shannon defined entropy as follows:

$$H = \sum_{i=1}^{k} P(O_i) \times log_2(P(O_i)) \tag{2}$$

In Eq. (2) k is the number of possible outcomes and $P(O_i)$ is the probability of outcome i. Entropy increases as the uncertainty relating to outcome increases. An entropy of 0 indicates that the outcome is certain: there is one outcome with a probability of 1 and all other outcomes have a probability of 0. The maximum value that entropy can take is dependent on the relationship between the log base that is used and the number of outcomes. When a log of base 2 is used the maximum entropy for scenarios with 2 outcomes is 1, however for scenarios with more than 2 outcomes the maximum entropy calculated using log base 2 can be greater than 1.

For our analysis, the basic definition of an outcome is a data point of a particular type ending up in a given cluster. So there are k outcomes when there are k clusters and we use log base 2 in all our calculations. We analyse the entropy across clusters along two dimensions: (i) the entropy of anomalies across clusters, and (ii) the entropy of points within a time series across clusters.

7.1. Entropy of anomalies across clusters

Our motivation for analysing the entropy of anomalies across clusters is the intuition that the lower the entropy of anomalies across clusters the more the clustering process is sorting anomalies into specific clusters, or in other words the more the clustering process is focused on doing anomaly detection (this is the outcome illustrated in Fig. 2). Conversely, as the entropy for anomalies increases the more the clustering process is identifying subdomains based on distinctions other than anomaly versus not-anomaly (for example, resulting in an outcome more similar to that illustrated in Fig. 3).

In calculating the entropy of anomalies across clusters an outcome is defined as an anomaly being allocated to a specific cluster. For example, if $k = 2$ there are two clusters and the entropy of anomalies across clusters would be

$$H(anomalies, k = 2) = -P(C_1) \times log_2(P(C_1)) + P(C_2) \times log_2(P(C_2)) \tag{3}$$

$$P(C_i) = \frac{\text{number of anomalies in } C_i}{\text{total number of anomalies in dataset}} \tag{4}$$

We are primarily interested in understanding the interaction between the type of normalisation applied to the data and the structure of the resulting subdomains. Consequently, we wish to control for the number of clusters (the value of k) in our analysis and to compare the composition of the domains across values of k. However, Shannon's measure of entropy is known to be sensitive to the number of outcomes. This means that in this scenario it is expected that as the number of outcomes increases (i.e., as k increases) then the entropy for the distribution of anomalies will increase. To be able to compare the entropy of anomalies across different values of k, we normalised the entropy for each value of k by the maximum entropy for k outcomes

$$MaxEntropy(k) = -k \times (P(\frac{1}{k}) \times log_2(P(\frac{1}{k}))) \tag{5}$$

Consequently, the normalised entropy for anomalies over k clusters is

$$NormalisedEnt(anomalies, k = i) = \frac{H(anomalies, k = i)}{MaxEntropy(k = i)} \tag{6}$$

Finally, k-means clustering is stochastic (non-deterministic) meaning that each time a k-means clustering process is executed the composition of the resulting clusters will be different (in some cases the difference can be very large). To control for this, we have used the k-means++ algorithm to initialise the centres of k-means clustering [21]. This ensures that the initialised centres are consistent across multiple runs on the same data.

Algorithm 2 lists the process followed to analyse the entropy of the anomalies for different types of normalisation.

Algorithm 2 Experiment 2 Methodology.

```
for each dataset do
    for each type of normalisation do
        for each value of k in 2,...,10 do
            Run k-means++ clustering
            Calculate the normalised entropy of anomalies
            across the clusters
        Calculate the average normalised entropy across
        the values of k
```

Table 3 lists the mean normalised entropy for anomalies for each dataset and normalisation type. There is a rank order for the entropies that holds for most of the datasets where the entropy for anomalies is lowest when no normalisation is applied and highest when normalisation is applied after feature extraction: $H_{NoNormalisation} < H_{BeforeFeatureExtraction} < H_{AfterFeatureExtraction}$. Low entropy indicates that the anomalies tend to be clustered together (similar to the scenario in Fig. 2) whereas high entropy indicates that anomalies tend to be spread across the clusters (tending towards Fig. 3 scenario).

Table 3
The mean normalised entropy for anomalies for each dataset across clusters averaged over different values of k.

Normalisation	Mean normalised entropy by dataset					
	AWS	Huawei	IOPS	Twitter	Yahoo	
					Artificial	Real
After feat. Ext.	0.934	0.909	0.808	0.919	0.529	0.919
Before feat. Ext.	0.686	0.429	0.468	0.616	0.572	0.877
No normalisation	0.214	0.639	0.112	0.038	0.434	0.117

Table 4
Mean normalised entropy describing the distribution of points from files across different clusters (averaged over different values of k).

Normalisation	Mean normalised entropy by dataset					
	AWS	Huawei	IOPS	Twitter	Yahoo	
					Artificial	Real
After feat. Ext.	0.744	0.764	0.786	0.856	0.899	0.864
Before feat. Ext.	0.393	0.442	0.372	0.576	0.563	0.482
No normalisation	0.085	0.153	0.015	0.013	0.039	0.010

7.2. Entropy of points within a time series across clusters

Cloud services are composed of multiple components, and cloud service datasets combine monitoring data from across these components. Furthermore, the data stream of monitoring data is often broken up into multiple files, with each file containing a separate time series of monitoring data from a single component of the cloud service. Given this structure within the datasets, it is to be expected that if a clustering process is focused on distinguishing anomalies from non-anomalies it is likely to distribute data points from the same time series (i.e., file) across multiple clusters (as illustrated in Fig. 2), whereas if the clustering process is identifying subdomains that integrate data from components with similar underlying distributions (the scenario illustrated in Fig. 3) it is to be expected that the clustering process should generally keep all the data points from a given time series together in one cluster (or a small number of clusters). As a result, in this experiment, we extend the analysis of the structure of the subdomains generated by the clustering process as a result of different types of normalisation to consider the average entropy of the points from times series (files) across clusters. This analysis is similar in structure to that based on the entropy of anomalies, and Algorithm 3 lists the steps in the process used to analyse the entropy of the data points from time series under different types of normalisation.

Algorithm 3 Experiment 3 Methodology.

```
for each dataset do
    for each type of normalisation do
        for each value of k in 2,...,10 do
            Run k-means++ clustering
            for each file f in the dataset do
                Calculate the normalised entropy of points in f
                across clusters
            Calculate the average entropy for points across
            the files in the dataset
```

Table 4 lists the results generated using this algorithm where the results are for each dataset and type of data normalisation. The rank order of entropies by normalisation types for most of the datasets follows a similar pattern to the rank order for the entropies of anomalies for the different normalisation types: the entropy for points from files is lowest when no normalisation is applied and highest when normalisation is applied after feature extraction: $H_{NoNormalisation} < H_{BeforeFeatureExtraction} < H_{AfterFeatureExtraction}$. This result is somewhat surprising as we had expected that for a given type of normalisation anomalies and normal points would be distributed differently. However, the results of this analysis of the entropy of points from a file taken together with the entropy of anomalies indicate that when no normalisation is applied anomalies are clustered together and the majority of points from a file are also clustered together. However, this does not mean that the anomalies within a file end up in the same cluster as the normal data points in a file. It is more likely that anomalies end up in clusters that just contain anomalies and the normal points in a file get bundled with normal points from other files. In contrast to this, when normalisation is applied after feature extraction both anomalies and normal points within a file have relatively high entropy indicating that they are both distributed across multiple clusters. This is particularly interesting as the results from Experiment 1 indicate that this process of distributing both anomalies and normal points across clusters (something that is a blend of the scenarios shown in Figs. 2 and 3) results in the best performance.

8. Experiment 3: Performance on merged datasets with transfer learning

In this section, we report experiments that use a collection of labelled *source* datasets to train an anomaly detection model for a *target* dataset, and so in these experiments, we perform transfer learning. To do this we treat each of the 6 datasets as a target dataset in turn and a new *multi-source dataset* is created by merging the other five datasets. This process of data handling resulted in the following transfer combinations (source → target): (1) Non-Aws → Aws, (2) Non-Huawei → Huawei, (3) Non-IOPS → IOPS, (4) Non-Twitter → Twitter, (5) Non-Yahoo$_{Artificial}$ → Yahoo$_{Artificial}$, and (6) Non-Yahoo$_{Real}$ → Yahoo$_{Real}$. The IOPS dataset is significantly larger than the rest of the datasets, so a 5% stratified sample, ensuring the same proportion of anomalies is present, was taken when it was used as a source dataset. The full IOPS dataset is used when it is the target dataset.

We report results from two versions of an experiment, the difference between the versions being that in the first experiment the definition of the sub-domains is done by running a clustering processing on the source dataset whereas in the second experiment the definition of the sub-domains is done by running a clustering process on the target dataset. From the results of the first experiment (clustering done on the source dataset), we assess (1) model performance on the test set of the source data (Section 8.1) and (2) model performance on the test set of the target data (Section 8.2). From the results of the second experiment (clustering done on the target dataset), we assess model performance on the test set of the target data when clustering is performed on the target instead of the source (Section 8.3). Also by comparing the results of the two experiments we investigate the impact of the number of clusters (Section 8.4).

A prerequisite of successful transfer learning is that the source and target datasets are *similar*. To address this we introduce a domain adaptation step after clustering (i.e., sub-domain definition) but before model training. In our experiments, we train a separate anomaly detection model for each sub-domain. The anomaly detection model for each sub-domain is trained using the source data points assigned to that sub-domain, and the target data points assigned to a sub-domain are classified by this model. Consequently, within each sub-domain, we wish to align the source data points to the target data points as this will likely improve transfer performance by making the marginal distribution of the training data (source data points) more similar to that of the test data (target data points). Zhang et al. [10] used an unsupervised domain adaption method developed by Sun et al. [24] called CORAL to increase the similarity between the set of data points in a source sub-domain and the set of target data points in the same sub-domain, and we use the same method in our experiments. However, there is a possibility of an interaction between the CORAL domain adaptation method and one or more of the preceding steps in our processing pipeline (feature set (Catch24), normalisation method (After Feature Extraction, Before Feature Extraction, or No Normalisation), or clustering method (k-means++)) that may inadvertently bias our results. To control for this we also report results when no domain adaptation is applied and when an alternative unsupervised domain adaptation method known as Subspace Alignment [25] is applied. Consequently, each analysis presented in this section draws on three sets of results, one for each form of domain adaptation: none, CORAL, and subspace alignment. However, for space considerations, we will limit the analysis of the results to F1. CORAL has the advantage of not requiring any hyper-parameters to be set. The subspace alignment method integrates does have one hyper-parameter which specifies the size of the subspaces. For our experiments, we use the default value of the size of the input dimensionality, which in our case is 24.

Algorithm 4 lists the process used in this experiment when the definition of the sub-domains is achieved by clustering on the source dataset. The only change in the second version of the experiment is that k-means++ is applied to the target dataset. Note that in this experiment the number of clusters has been extended to include a single cluster (i.e., k ranges from 1 to 10).

Algorithm 4 Experiment 4 Methodology.

```
Select a dataset as the target, merge other datasets to form the source dataset
Domain adaptation methods = {none, CORAL, Subspace Alignment}
for each type of normalisation do
    for k in 1,...,10 do
        Fit k-means++ model on the source dataset to identify k clusters
        Use stratified 5-fold sampling to form training and test sets for both source and target datasets
        for each fold in 5-fold cross validation do
            Apply k-clustering model to source and target training/test data
            for each domain adaptation method do
                for each of k clusters do
                    Apply domain adaptation to source training and test data in the cluster to form adapted source training and test sets
                    Fit anomaly detection model on the adapted source training set
                    Evaluate model on the adapted source and target test sets
Calculate an average F1 for each combination of domain adaptation and normalisation across all folds, and all values of k and all datasets for the source and target test sets
```

8.1. Model performance on merged source datasets (no transfer)

The experimental procedure we follow means that for each transfer pair of datasets we have two test sets, the first is a fold from the merged source datasets that have been adapted by applying a domain adaptation transform generated for each cluster to the data points assigned to that cluster. The second is the sample of the target dataset that was not used in the domain adaptation process. In this section, we present the performance of our models on the first of these test sets—the adapted source test set. These results do not involve transfer learning because both the training and test sets are sampled from the same merged datasets. Consequently, these results enable us to assess whether there is an interaction between any of the normalisation processes and the domain adaptation methods which might disrupt the distribution of the multi-domain source datasets. Table 5 presents for each of the three domain adaptation conditions and each normalisation type the mean F1 along with its 95% confidence interval. The set of values used to calculate each of these averages was the result of combining 6 datasets with 5 folds per dataset and 10 iterations of clustering per fold (k in the range 1 through 10) resulting in 300 values.

The best overall F1 is obtained when no domain adaptation is applied and using normalisation after feature extraction F1= 0.9164, and more generally better performance is obtained when no domain adaptation is applied. The higher performance for the no domain adapted condition relative to either of the domain-adapted conditions is statistically significant across all the normalisation conditions. This can be explained by the fact that in this experiment both the training and test sets are separate samples from the same original distribution (the merged source datasets) and so applying a domain adaptation method to transform these samples to be more similar to the training sample from the target distribution may make these two source samples less similar to each other. Comparing the results between the two domain adaptation conditions, in general (excluding F1 under the no normalisation condition) using CORAL for domain adaptation results in higher performance as compared with using subspace alignment and the difference in performance is statistically significant.

Switching the focus to the effect of normalisation on performance, both no domain adaption and CORAL normalisation after feature extraction performs the best and both normalisation methods perform better than no normalisation. Furthermore, these differences in performance are statistically significant both between the two normalisation methods and between the two normalisation methods and no normalisation. However, when subspace alignment domain adaptation is applied the results are mixed across the normalisation methods, normalisation after feature extraction has the lowest F1 and normalisation before feature extraction has the best F1 but the difference in performance between normalisation before feature extraction and no normalisation is not statistically significant. Overall we interpret these results as suggesting that normalisation after feature extraction generally produces the best results (best F1 by a statistically significant margin for both the no domain adaptation and CORAL conditions).

8.2. Model performance on target datasets (transfer learning)

Table 6 presents the average F1 scores for our models when evaluated on the test sets from the target domain datasets of each domain transfer pair when different domain adaptation methods are applied to the source training data in each cluster. As was the case in the previous experiment the best overall F1 is obtained when no domain adaptation is applied and using normalisation after feature extraction F1= 0.4052. However, as this max F1 highlights there is a significant drop in performance when compared with the performance on the adapted source domain test sets, this is not surprising as this experiment now involves transfer learning and so the task is intrinsically more difficult. Furthermore, although the max overall F1 score is obtained when no domain adaptation method is used, domain adaptation does appear to help. In particular, CORAL domain adaptation in combination with normalisation after feature extraction achieves the next highest overall F1 score of 0.2943, and in combination with normalisation before feature extraction achieves the third highest overall F1 score of 0.2414.

Table 5
For each domain adaptation condition (none, CORAL and subspace alignment) the mean F1 for each type of normalisation is calculated across the six source test sets (one test set from each of the transfer pairs of datasets) along with the calculated to 95% confidence interval for each metric mean. Population size 300.

No domain adaptation	Average F1	Confidence (Error margin)	Range
After feature extraction	0.9164	±0.0016	(0.9148, 0.9179)
Before feature extraction	0.9010	±0.0018	(0.8992, 0.9029)
No normalisation	0.8945	±0.0018	(0.8927, 0.8963)
CORAL	Average F1	Confidence (Error margin)	Range
After feature extraction	0.8670	±0.0033	(0.8637, 0.8703)
Before feature extraction	0.8521	±0.0032	(0.8489, 0.8852)
No normalisation	0.7375	±0.0060	(0.7315, 0.7435)
Subspace alignment	Average F1	Confidence (Error margin)	Range
After feature extraction	0.7390	± 0.0065	(0.7324, 0.7455)
Before feature extraction	0.8068	± 0.0034	(0.8034, 0.8102)
No normalisation	0.8036	± 0.0036	(0.8, 0.8071)

Table 6
For each domain adaptation condition (none, CORAL and subspace alignment) the mean F1 for each type of normalisation is calculated across the six target domain test sets (one test set from each of the transfer pairs of datasets) along with the calculated to 95% confidence interval for each metric mean. Population size 300.

No domain adaptation	Average F1	Confidence (Error margin)	Range
After feature extraction	0.4052	± 0.0262	(0.379, 0.4314)
Before feature extraction	0.2225	± 0.0050	(0.2176, 0.2275)
No normalisation	0.0947	± 0.0091	(0.0857, 0.1038)
CORAL	Average F1	Confidence (Error margin)	Range
After feature extraction	0.2943	±0.0163	(0.2780, 0.3106)
Before feature extraction	0.2414	±0.0108	(0.2306, 0.2522)
No normalisation	0.1502	±0.0092	(0.1411, 0.1594)
Subspace alignment	Average F1	Confidence (Error margin)	Range
After feature extraction	0.2055	± 0.0121	(0.1934, 0.2175)
Before feature extraction	0.1541	± 0.0077	(0.1464, 0.1618)
No normalisation	0.1147	± 0.0077	(0.107, 0.1224)

Finally, in terms of normalisation strategies, across the three domain adaptation conditions, normalisation after feature extraction is the best performing by a statistically significant margin compared to both normalisation before feature extraction and no normalisation. Furthermore, when compared to performance on the adapted source test sets (i.e., the results reported in Section 8.1) the difference between F1 scores with normalisation after feature extraction and before feature extraction is wider for each of the three domain adaptation conditions. This may suggest that normalisation after feature extraction better identifies subdomains allowing transfer learning to work better.

8.3. *Model performance on target datasets: Clustering on target*

The majority of research on transfer learning that deals with multi-source transfer learning focuses on identifying sub-domains within the source dataset. However, multiple sub-domains may exist within the target domain. To explore this we ran a second version of our experiment where we ran the clustering (sub-domain definition) process on the training portion of the target dataset, rather than on the source dataset. Apart from this modification the rest of the experimental procedure was the same as that described in Algorithm 4. Table 7 presents the results for this version of the experiment.

Table 7
For each domain adaptation condition (none, CORAL and subspace alignment) the mean F1 for each type of normalisation calculated across the six target domain test sets (one test set from each of the transfer pairs of datasets) when clustering (sub-domain definition) is based on the training samples from the target domain, along with the calculated to 95% confidence interval for each metric mean. Population size 300.

No domain adaptation	Average F1	Confidence (Error margin)	Range
After feature extraction	0.4002	± 0.0284	(0.3718, 0.4286)
Before feature extraction	0.2226	±0.0050	(0.2176, 0.2275)
No normalisation	0.1069	±0.0097	(0.0972, 0.1166)
CORAL	Average F1	Confidence (Error margin)	Range
After feature extraction	0.4136	±0.0247	(0.3889, 0.4383)
Before feature extraction	0.2643	±0.0118	(0.2525, 0.2761)
No normalisation	0.1286	±0.0108	(0.1179, 0.1394)
Subspace alignment	Average F1	Confidence (Error margin)	Range
After feature extraction	0.3067	±0.0250	(0.2817, 0.3317)
Before feature extraction	0.1393	±0.0068	(0.1325, 0.1461)
No normalisation	0.1325	±0.0073	(0.1252, 0.1398)

Table 7 shows similar results to Table 6 with normalisation after feature extraction remaining the best performing strategy, and there are significant differences between the respective normalisation strategies, with normalisation after feature extraction resulting in significantly better performance. Furthermore, the beneficial effect of adapting the source training data to the target domain is particularly apparent in these results with the best performance for all three types of normalisation being achieved when some form of domain adaptation is applied (as compared with both Tables 5 and 6 where the best performance was obtained when no domain adaptation was applied). This suggests that there is a beneficial interaction between clustering on the target for sub-domain definition and domain adaptation to the target. Moreover, clustering on the target also appears to improve overall performance when compared to clustering on the source dataset. The overall max F1 score in Table 7 is 0.4136, as compared with an overall max F1 of 0.4052 in Table 6. Admittedly the margin of difference between these two max F1 scores is not statistically significant at the 95% confidence level, however comparing across the normalisation strategies and the two tables there does appear to be a trend where clustering on the target leads to improved performance. Across the 9 F1 scores reported in each table in six instances the F1 score obtained by clustering on the target is higher than the corresponding F1 obtained when clustering on the source, and in four of these instances the margin of difference is statistically significant at the 95% confidence level. This result is somewhat surprising as we had expected that clustering on the larger merged source datasets would identify better subdomains, assuming the large source datasets would contain more distinct subdomains. However, it is possible that clustering on the merged source dataset may identify subdomains that are not present in the target dataset, resulting in unused training data. Conversely, clustering on the target data may result in all clusters being relevant to the target data and hence utilising more of the source data.

8.4. *Model performance with number of clusters*

In this experiment, we look at the best performing normalisation strategy and domain adaptation combination (after feature extraction and CORAL) and investigate how the number of clusters impacts model performance. We do this analysis for both clustering on the source and the target training data. Table 8 lists for clustering on the training sample from the source domain

Table 8
Using the clustering on training sample from the merged source domain datasets to define relevant sub-domains: For each transfer pair of datasets the max F1 achieved across the different values of k used in clustering, the k value that resulted in the max F1 value, the average F1 and standard deviation across values of k.

Cluster on source Test on target	Max F1	Best k	Average F1	Standard deviation
Non-AWS→AWS	0.2990	1	0.2242	0.0296
Non-Huawei→Huawei	0.3290	1	0.2667	0.0418
Non-IOPS→IOPS	0.3406	6	0.3276	0.0102
Non-Twitter→Twitter	0.1291	4	0.1231	0.0050
Non-$\text{Yahoo}_{\text{Artificial}}$ → $\text{Yahoo}_{\text{Artificial}}$	0.7418	1	0.2887	0.1632
Non-$\text{Yahoo}_{\text{Real}}$ → $\text{Yahoo}_{\text{Real}}$	0.5818	1	0.5364	0.0228

Table 9
Using the clustering on training sample from the target domain dataset to define relevant sub-domains: For each transfer pair of datasets the max F1 achieved across the different values of k used in clustering, the k value that resulted in the max F1 value, the average F1 and standard deviation across values of k.

Cluster on source Test on target	Max F1	Best k	Average F1	Standard deviation
Non-AWS→AWS	0.3080	1	0.2770	0.0236
Non-Huawei→Huawei	0.4845	9	0.4055	0.0642
Non-IOPS→IOPS	0.3468	3	0.3347	0.0098
Non-Twitter→Twitter	0.1346	9	0.1283	0.0040
Non-$\text{Yahoo}_{\text{Artificial}}$ → $\text{Yahoo}_{\text{Artificial}}$	0.8429	5	0.7619	0.1597
Non-$\text{Yahoo}_{\text{Real}}$ → $\text{Yahoo}_{\text{Real}}$	0.6061	6	0.5748	0.0204

for each transfer dataset pair the max F1 value achieved across the different values of k used in clustering, the corresponding k value that resulted in the max value, the average F1 and standard deviation calculated across the different values of k for that clustering strategy and transfer pair. Table 9 lists the same data for when clustering is applied to the sample of training data from the target dataset.

The results in Tables 8 and 9 reinforce that there are benefits in clustering on the target. Model performance improves not only the average performance but also the max F1. When clustering on the source data most of the time the best performance is without clustering or one cluster. The two exceptions IOPS and Twitter have the lowest variation in the F1, and so clustering has the least effect on their performance. The fact that clustering appears to have a stronger impact when applied to the target dataset (i.e., the value of k affects performance and more often than not the best performance occurs with k>1) indicates that clustering on the target identifies more relevant sub-domains.

9. Conclusions

This paper explored the interaction between data normalisation and clustering in the definition of sub-domains within multi-source transfer learning for time series anomaly detection. Our results suggest that normalisation after feature extraction gives the best performance. Furthermore, our analysis found that normalisation after feature extraction tends to result in the highest entropy in terms of both the distribution of anomalies within a dataset but also the distribution of points from a file. Taken together these results indicate that a normalisation and clustering combination that results in both anomalies within the dataset and points from files being distributed across clusters tends to result in better performance. This may indicate that normalisation after feature extraction results in clustering identifying sub-domains/components based on underlying (latent) similarities across points within a dataset as opposed to distinctions based on either normal versus anomalous data points or the data process that generates data points.

Concerning transfer learning our results suggest that when performing transfer learning normalising before clustering improves model performance (either before or after feature extraction) compared to not normalising, and again we find that normalisation after feature extraction gives the best performance and that this is statistically significant in terms of F1. Furthermore, of the two unsupervised domain adaptation methods we used in our experiments we find CORAL is more effective than subspace alignment. Perhaps most surprisingly, however, we find that clustering on the target dataset and identifying subdomains in the target data can result in improved model performance compared to the standard approach of identifying sub-domain in the source dataset. The best overall performance for multi-source transfer learning is obtained using normalisation after feature selection combined with clustering on the target and applying CORAL for domain adaptation. However, a lower but statistically similar performance is also obtained if we do no domain adaptation and cluster on either the source or the target, so long as we apply normalisation after feature extraction.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was funded by Huawei Research Ireland, and by the ADAPT Centre which is funded under the SFI Research Centres Programme (Grant 13/RC/2106_P2) and is co-funded under the European Regional Development Funds.

References

[1] N. Görnitz, M. Kloft, K. Rieck, U. Brefeld, Toward supervised anomaly detection, J. Artificial Intelligence Res. 46 (2013) 235–262.
[2] J.D. Kelleher, Deep Learning, MIT Press, 2019.
[3] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2009) 1345–1359.
[4] K. Weiss, T.M. Khoshgoftaar, D. Wang, A survey of transfer learning, J. Big Data 3 (1) (2016) 1–40.
[5] M.T. Rosenstein, Z. Marx, L.P. Kaelbling, T.G. Dietterich, To transfer or not to transfer, in: NIPS 2005 Workshop on Transfer Learning, Vol. 898, 2005, pp. 1–4.
[6] W. Zhang, L. Deng, L. Zhang, D. Wu, Overcoming negative transfer: A survey, 2020, arXiv preprint arXiv:2009.00909.
[7] S. Christodoulidis, M. Anthimopoulos, L. Ebner, A. Christe, S. Mougiakakou, Multisource transfer learning with convolutional neural networks for lung pattern analysis, IEEE J. Biomed. Health Inf. 21 (1) (2016) 76–84.
[8] C. Yi, Y. Xu, H. Yu, Y. Yan, Y. Liu, Multi-component transfer metric learning for handling unrelated source domain samples, Knowl.-Based Syst. 203 (2020) 106132, http://dx.doi.org/10.1016/j.knosys.2020.106132, URL https://www.sciencedirect.com/science/article/pii/S0950705120303877.
[9] S. Moon, J.G. Carbonell, Completely heterogeneous transfer learning with attention-what and what not to transfer, in: IJCAI, Vol. 1, no. 1, 2017 pp. 1–2.
[10] X. Zhang, Q. Lin, Y. Xu, S. Qin, H. Zhang, B. Qiao, Y. Dang, X. Yang, Q. Cheng, M. Chintalapati, Y. Wu, K. Hsieh, K. Sui, X. Meng, Y. Xu, W. Zhang, F. Shen, D. Zhang, Cross-dataset time series anomaly detection for cloud systems, in: 2019 USENIX Annual Technical Conference, USENIX ATC 19, USENIX Association, Renton, WA, 2019, pp. 1063–1076, URL https://www.usenix.org/conference/atc19/presentation/zhang-xu.
[11] T.-Y. Kim, S.-B. Cho, Web traffic anomaly detection using C-LSTM neural networks, Expert Syst. Appl. 106 (2018) 66–76.
[12] M. Ahmed, A.N. Mahmood, J. Hu, A survey of network anomaly detection techniques, J. Netw. Comput. Appl. 60 (2016) 19–31.
[13] G. Münz, S. Li, G. Carle, Traffic anomaly detection using k-means clustering, in: GI/ITG Workshop MMBnet, 2007, pp. 13–14.
[14] I. Syarif, A. Prugel-Bennett, G. Wills, Unsupervised clustering approach for network anomaly detection, in: International Conference on Networked Digital Technologies, Springer, 2012, pp. 135–145.

[15] B.D. Fulcher, N.S. Jones, hctsa: A computational framework for automated time-series phenotyping using massive feature extraction, Cell Syst. 5 (5) (2017) 527–531.

[16] C.H. Lubba, S.S. Sethi, P. Knaute, S.R. Schultz, B.D. Fulcher, N.S. Jones, catch22: Canonical time-series characteristics, Data Min. Knowl. Discov. 33 (6) (2019) 1821–1852.

[17] R. Agrahari, M. Nicholson, C. Conran, H. Assem, J.D. Kelleher, Assessing feature representations for instance-based cross-domain anomaly detection in cloud services univariate time series data, IoT 3 (2022) 123–144, http://dx.doi.org/10.3390/iot3010008.

[18] A. Lavin, S. Ahmad, Evaluating real-time anomaly detection algorithms–The numenta anomaly benchmark, in: 2015 IEEE 14th International Conference on Machine Learning and Applications, ICMLA, IEEE, 2015, pp. 38–44.

[19] N. Laptev, S. Amizadeh, I. Flint, Generic and scalable framework for automated time-series anomaly detection, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 1939–1947.

[20] M. Vlachos, P. Yu, V. Castelli, On periodicity detection and structural periodic similarity, in: Proceedings of the 2005 SIAM International Conference on Data Mining, SIAM, 2005, pp. 449–460.

[21] D. Arthur, S. Vassilvitskii, k-means++: The Advantages of Careful Seeding, Tech. Rep., Stanford, 2006.

[22] C.C. Aggarwal, Data Mining: The Textbook, Springer, 2015.

[23] R. Wu, E. Keogh, Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress, IEEE Trans. Knowl. Data Eng. (2021) 1, http://dx.doi.org/10.1109/TKDE.2021.3112126.

[24] B. Sun, J. Feng, K. Saenko, Return of frustratingly easy domain adaptation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 30, no. 1, 2016.

[25] B. Fernando, A. Habrard, M. Sebban, T. Tuytelaars, Subspace alignment for domain adaptation, 2014, CoRR, abs/1409.5241, arXiv:1409.5241.