Pattern Recognition 58 (2016) 121–134

Contents lists available at ScienceDirect

Pattern Recognition

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

High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning

Sarah M. Erfani*, Sutharshan Rajasegarar[1], Shanika Karunasekera, Christopher Leckie

NICTA Victoria Research Laboratory, Department of Computing and Information Systems, Room 7.14, Dough MacDonell Building, The University of Melbourne, VIC 3010, Australia

ARTICLE INFO

Article history:
Received 8 April 2015
Received in revised form
29 February 2016
Accepted 28 March 2016
Available online 14 April 2016

Keywords:
Anomaly detection
Outlier detection
High-dimensional data
Deep belief net
Deep learning
One-class SVM
Feature extraction

ABSTRACT

High-dimensional problem domains pose significant challenges for anomaly detection. The presence of irrelevant features can conceal the presence of anomalies. This problem, known as the '*curse of dimensionality*', is an obstacle for many anomaly detection techniques. Building a robust anomaly detection model for use in high-dimensional spaces requires the combination of an unsupervised feature extractor and an anomaly detector. While one-class support vector machines are effective at producing decision surfaces from well-behaved feature vectors, they can be inefficient at modelling the variation in large, high-dimensional datasets. Architectures such as deep belief networks (DBNs) are a promising technique for learning robust features. We present a hybrid model where an unsupervised DBN is trained to extract generic underlying features, and a one-class SVM is trained from the features learned by the DBN. Since a linear kernel can be substituted for nonlinear ones in our hybrid model without loss of accuracy, our model is scalable and computationally efficient. The experimental results show that our proposed model yields comparable anomaly detection performance with a deep autoencoder, while reducing its training and testing time by a factor of 3 and 1000, respectively.

1. Introduction

The growth in pervasive network infrastructure, such as the Internet of Things (IoT), is enabling a wide range of physical objects and environments to be monitored in fine spatial and temporal detail [1,2]. A key challenge in the development of IoT applications is how to model and interpret the large volumes of high-dimensional data that are generated in such domains [2]. Further, the lack of ground truth (labels) in the data that are collected from large-scale networks in the IoT require unsupervised algorithms to process the data. Anomaly detection aims to detect unusual behaviours caused by either faulty devices or events of interest in the monitoring environment, and thus is of great importance in IoT applications. However, a major challenge for anomaly detection in such domains is how to cope with noisy, large-scale datasets [3-6]. In this work we address this challenge by proposing an unsupervised hybrid architecture for anomaly detection in large-scale high-dimensional problem domains.

A core challenge for anomaly detection that distinguishes it from other classification problems is that in many cases anomaly detection algorithms should be trained with unlabelled records, i.e., trained in an unsupervised manner. Obtaining a large training set of clean and labelled data is often a labour and time intensive task. Moreover, anomaly detection becomes more challenging when applied to high-dimensional datasets that contain a large number of records. Many of the available methods for identifying anomalies assume small datasets with low numbers of features.

High-dimensional datasets pose a challenge for anomaly detection due to the following factors [7]: (i) *Exponential search space* — The number of potential feature subspaces grows exponentially with increasing input dimensionality, resulting in an exponential search space. (ii) *Data-snooping bias* — Every point in a high-dimensional space appears as an anomaly. Given enough alternative subspaces, at least one feature subspace can be found for each point such that it appears as an anomaly. (iii) *Irrelevant features* — A high proportion of irrelevant features effectively creates noise in the input data, which masks the true anomalies. The challenge is to choose a subspace of the data that highlights the relevant attributes.

Our objective is to find a large-scale, high-dimensional anomaly detection algorithm that is *robust*, i.e., generates an accurate model for data drawn from a wide range of probability distributions, and is not unduly affected by small departures from the trained model. In addition, it is desirable that the algorithm be efficient in terms

* Corresponding author. Tel. +61 3 83440366.
E-mail address: sarah.erfani@unimelb.edu.au (S.M. Erfani).

[1] This author is now with: School of Information Technology, Deakin University, Australia

http://dx.doi.org/10.1016/j.patcog.2016.03.028
0031-3203/© 2016 Elsevier Ltd. All rights reserved.

of *time complexity*, *memory complexity* and the *required number of labelled records*.

One-class Support Vector Machines (1SVMs) [8–10] are a popular technique for unsupervised anomaly detection. Generally, they aim to model the underlying distribution of normal data while being insensitive to noise or anomalies in the training records. A kernel function implicitly maps the input space to a higher dimensional feature space to make a clearer separation between normal and anomalous data. When properly applied, in principle a kernel-based method is able to model any non-linear pattern of normal behaviour. For clarity in the rest of the paper, the notation of 1SVM is used to denote (an unsupervised) one-class SVM; *l*SVMs — short for labeled SVM — to denote (supervised) binary and multi-class SVM classifiers; and SVMs when both 1SVMs and *l*SVMs are considered.

SVMs are theoretically appealing for the following reasons [11,12]: they provide good generalisation when the parameters are appropriately configured, even if the training set has some bias; they deliver a unique solution, since the loss function is convex; and in principal they can model any training set, when an appropriate kernel is chosen.

In practice, however, training SVMs is memory and time intensive. SVMs are non-parametric learning models, whose complexity grows quadratically with the number of records [13]. They are best suited to small datasets with many features, and so far large-scale training on high-dimensional records (e.g., $10^6 \times 10^4$) has been limited with SVMs [14]. Large numbers of input features result in the curse of dimensionality phenomenon, which causes the generalisation error of shallow architectures (discussed in Section 2.1), such as SVMs, to increase with the number of irrelevant and redundant features. The curse of dimensionality implies that to obtain good generalisation, the number of training samples must grow exponentially with the number of features [14,4,15]. Furthermore, shallow architectures have practical limitations for efficient representation of certain types of function families [16]. To avoid these major issues, it is essential to generate a model that can capture the large degree of variation that occurs in the underlying data pattern, without having to enumerate all of them. Therefore, a compact representation of the data that captures most of the variation can alleviate the curse of dimensionality as well as reducing the computational complexity of the algorithm [16,17].

An alternative class of classification algorithms that have emerged in recent years are Deep Belief Nets (DBNs), which have been proposed as a multi-class classifier and dimensionality reduction tool [18–20]. DBNs are multi-layer generative models that learn one layer of features at a time from unlabelled data. The extracted features are then treated as the input for training the next layer. This efficient, greedy learning can be followed by fine-tuning the weights to improve the generative or discriminative performance of the whole network.

DBNs have a deep architecture, composed of multiple layers of parameterised non-linear modules. There are a range of advantageous properties that have been identified for DBNs [16]: they can learn higher-level features that yield good classification accuracy; they are parametric models, whose training time scales linearly with the number of records; they can use unlabelled data to learn from complex and high-dimensional datasets.

A major limitation of DBNs is that their loss function is non-convex, therefore the model often converges on local minima and there is no guarantee that the global minimum will be found. In addition, DBN *classifiers* are semi-supervised algorithms, and require some labelled examples for discriminative fine-tuning, hence an unsupervised generative model of DBNs, known as autoencoders, are used for anomaly detection.

The open research problem we address is how to overcome the limitations of one-class SVM architectures on complex, high-dimensional datasets. We propose the use of DBNs as a feature reduction stage for one-class SVMs, to give a hybrid anomaly detection architecture. While a variety of feature reduction methods — i.e., feature selection and feature extraction methods — have been considered for SVMs (e.g., [21–25] — see [26] for a survey) none have studied the use of DBNs as a method for deep feature construction in the context of anomaly detection, i.e., with a one-class SVM. In this paper, we design and evaluate a new architecture for anomaly detection in high-dimensional domains. To the best of our knowledge, this is the first method proposed for combining DBNs with one-class SVMs to improve their performance for anomaly detection.

The contributions of this paper are two-fold. The performance of DBNs against one-class SVMs is evaluated for detecting anomalies in complex high-dimensional data. In contrast, the reported results in the literature from DBN classification performance only cover *multi-class* classification, e.g., [14,27–29]. A novel unsupervised anomaly detection model is also proposed, which combines the advantages of deep belief nets with one-class SVMs. In our proposed model an unsupervised DBN is trained to extract features that are reasonably insensitive to irrelevant variations in the input, and a 1SVM is trained on the feature vectors produced by the DBN. More specifically, for anomaly detection we show that computationally expensive non-linear kernel machines can be replaced by linear ones, when aggregated with a DBN. To the best of our knowledge, this is the first time these frameworks have been combined this way. The result of experiments conducted on several benchmark datasets demonstrate that our hybrid model yields significant performance improvements over the stand-alone systems. The combination of the hybrid DBN-1SVM avoids the complexity of non-linear kernel machines, and reaches the accuracy of a state-of-the-art autoencoder while considerably lowering its training and testing time.

The remainder of the paper is organised as follows. Section 2 begins with an introduction to deep architectures and their strengths and weaknesses compared to their shallow counterparts. Then it reviews some of the leading 1SVM methods, and motivates the requirements for the hybrid model by considering the shortcomings of SVMs for processing large datasets. Section 3 presents our proposed unsupervised anomaly detection approach DBN-1SVM. Section 4 describes the empirical analysis and provides a detailed statistical comparison of the performance of autoencoder, 1SVM and DBN-1SVM models on various real-world and synthetic datasets. It demonstrates the advantages of the DBN-1SVM architecture in terms of both accuracy and computational efficiency. Section 5 summarises the paper and outlines future research.

2. Background

2.1. Shallow and deep architectures

Classification techniques with shallow architectures typically comprise an input layer together with a single layer of processing. Kernel machines such as SVMs, for example, are a layer of kernel functions that are applied to the input, followed by a linear combination of the kernel outputs. In contrast, deep architectures are composed of several layers of nonlinear processing nodes. The widely used form of the latter type of architectures are multi-layer neural networks with multiple hidden layers.

While shallow architectures offer important advantages when optimising the parameters of the model, such as using convex loss functions, they suffer from limitations in terms of providing an efficient representation for certain types of function families. In particular, shallow models have difficulties in representing functions that capture dependencies in joint distributions, i.e., dependencies between multiple variables can be difficult to capture using a shallow architecture. They can also be inefficient in terms of the required number of computational elements and training examples. Consequently, a shallow architecture can result in a model that does not generalise well, unless trained with a very large number of examples and implemented with a sufficiently large number of nodes, thus increasing the computational resources required.

Deep architectures can include multiple layers of representation and abstraction to help model and generalise from complex datasets. This enables deep architectures to provide a compact representation for a much wider range of functions than is possible using shallow ones. In general, functions with a k-layer architecture can provide a compact representation using a number of hidden units that is polynomial in the number of input features, whereas a $(k-1)$-layer architecture requires an exponentially large number of hidden units. Furthermore, these units are generally organised in multiple layers so that many levels of computation can be composed. A thorough description of shallow and deep architectures is presented in [16], and the interested reader is referred to this paper for further information.

2.2. 1SVM, DBN and hybrid DBN-1SVM

One-class SVMs are widely used for anomaly detection. Their general approach is to implicitly map the data vectors from the *input space* to the *feature space* by means of a non-linear kernel function. The mapped vectors in the feature space are called *image vectors*. Then a smooth surface or boundary is found in the feature space that separates the image vectors into normal and anomalous measurements. By using a Mercer kernel function [30], the data vectors are implicitly mapped to a higher dimensional *inner product* space without any knowledge of the mapping function. The boundaries found in the implicit feature space usually yield a non-linear boundary in the input space [10].

Supervised SVMs (*l*SVMs — for labelled SVMs) are not always practical for anomaly detection, due to their need for labelled training data that identifies both normal and anomalous records in the dataset, or relying on “pure” normal data that are free of any anomalies. Such training datasets are expensive to compile in practice. Unsupervised anomaly detection approaches such as one-class SVMs overcome this challenge by constructing a non-linear model of normal behaviour, where data points that deviate from the normal model are identified as anomalies. Recently, several one-class SVM methods have been proposed for anomaly detection and some of the state-of-the-art one-class SVM formulations [8,9,31–35] are briefly described in the following.

Schölkopf et al. [31] proposed a hyperplane-based one-class SVM, where image vectors in the feature space are separated from the origin by a hyperplane with the largest possible margin. The vectors in the half space containing the origin are identified as anomalous. This scheme uses quadratic optimisation to fit a hyperplane. Campbell and Bennett [36] formulated a linear programming approach for the one-class SVM when used with a radial basis function (RBF) kernel. This formulation is based on attracting the hyperplane towards the average of the image data distribution, rather than minimising the maximum norm distance from the bounding hyperplane to the origin as in [31].

Tax and Duin [8] formulated the one-class SVM using a hypersphere, where a minimum radius hypersphere is fixed around the majority of the image vectors in the feature space. The data vectors that fall outside the hypersphere are identified as anomalies. The optimisation of the hypersphere formulation uses quadratic programming. Further, Tax and Duin have shown that the hyperplane-based one-class SVM becomes a special case of the (equivalent) hypersphere-based scheme when used with a radial basis kernel. Wang et al. [9] formulated the one-class SVM using hyperellipsoids with minimum effective radii around a majority of the image vectors in the feature space. This hyperellipsoidal formulation involves two phases. First, the image vectors are partitioned into a number of distinct clusters using Ward's algorithm [37]. Second, the image vectors in each cluster are fixed with a hyperellipsoid that encapsulates a majority of the image vectors in that cluster. The image vectors that do not fall within any of the hyperellipsoids are identified as anomalous. This problem is formulated as a second order cone programming optimisation problem, imposing a greater computational requirement than quadratic programming.

Laskov et al. [32] have observed the one-sidedness of the data distribution in many practical applications, and exploited this property to develop a special type of SVM called a one-class Quarter-Sphere SVM (QSSVM). This is extended from the hypersphere-based one-class SVM approach proposed by Tax and Duin [8]. The QSSVM finds a minimal radius hypersphere centred at the origin that encapsulates a majority of the image vectors in the feature space [32]. Data vectors that fall outside the quarter sphere are classified as anomalies. This problem is formulated as a linear programming problem. In [35], a distributed approach is presented using the QSSVM. In [33,34] a centred-hyperellipsoidal one-class SVM is presented based on a linear programming approach.

In contrast, deep belief nets are a relatively new type of multi-layer neural network. A DBN is trained in an unsupervised way to learn a hierarchical representation of the training data, which corresponds to a high-dimensional manifold. DBNs are trained one layer at a time, i.e., the latent variables at each layer are treated as the input for training the next layer. This efficient, greedy layer-wise training [38] can be followed by a stage of supervised fine-tuning, e.g., adding softmax or a logistic regression classifier, to improve the discriminative performance of the network. Semi-supervised DBNs are commonly used for multi-class classification. These methods have been shown to exhibit low complexity and high classification performance on complex datasets, e.g., for electroencephalography waveform clarification [39] and 3D object recognition [40], in comparison to other (shallow) classifiers such as SVMs.

DBNs have been demonstrated to be effective at learning invariant features from complex and high-dimensional datasets. While SVMs with their fixed kernel function can face difficulties in learning complicated invariances, but can learn robust decision surfaces when applied to well-behaved feature vectors. Given these complementary strengths, it is appealing to investigate the scope for using a hybrid model of these two architectures for anomaly detection.

Hybrid DBN-*l*SVM methods (i.e., a DBN in combination with labelled SVMs such as binary or multiclass SVMs) have been used in several application domains, such as object classification [14], text classification [27], music classification [29], face expression recognition [41], and speech separation [42]. These studies have observed that a hybrid model yields noticeable performance improvement over the stand-alone methods for *supervised* classification, but have not considered *unsupervised anomaly detection*. The next section presents our hybrid model that combines DBN and one-class SVM architectures, called DBN-1SVM, for *unsupervised* anomaly detection.

3. DBN-1SVM hybrid model

To benefit the complementary strengths of DBNs and kernel machines, we propose a novel *unsupervised* hybrid architecture

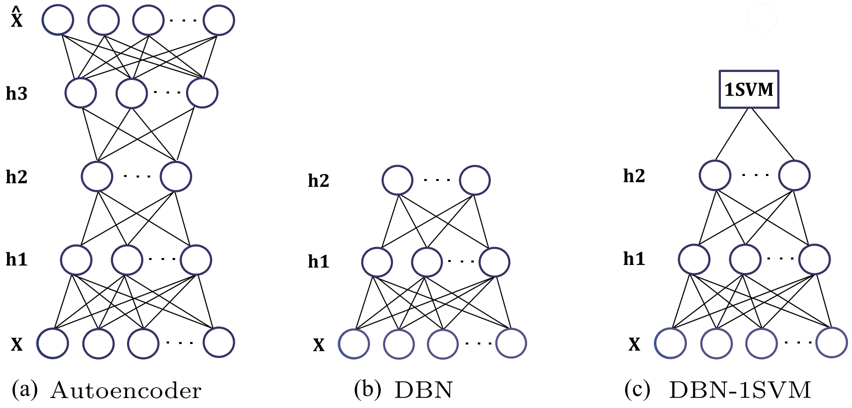

Fig. 1. Model architecture of AE, DBN and the proposed hybrid DBN-1SVM.

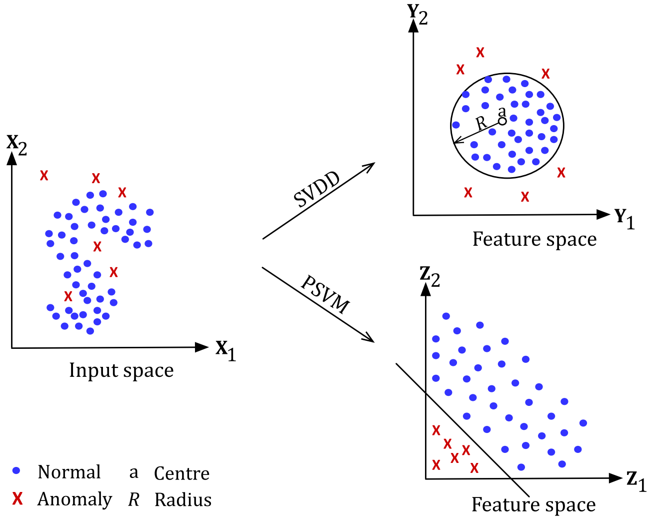

Fig. 2. Comparing SVDD and PSVM. The figure on the left shows a simple dataset in the input space. Normal records are represented with solid dots and anomalies with crosses. The figures on the right show the data projected to a higher dimensional space using two different 1SVM approaches.

DBN-1SVM (see Fig. 1), in which a DBN is trained to extract features that are relatively invariant to irrelevant variations in the input, so that the 1SVM can effectively separate the normal data from anomalies in the learned feature space. A DBN is trained as a non-linear dimensionality reduction algorithm to transform the high-dimensional data into a low-dimensional set of features [18]. The derived feature sets from the training samples form the input to train the one-class SVM. Subsequently, a hybrid of the generated models from these two algorithms constructs the ultimate anomaly detection model and can be used for testing. Our experiments show that this hybrid system not only improves the detection rate significantly, but also alleviates the computational complexity of training and testing.

In the following, we first describe the use of the DBN as a dimensionality reduction algorithm, and then elaborate on how the output of the DBN can be taken as input for a one-class SVM. For the explanation below, two of the most common 1SVM algorithms are chosen, a hypersphere-based 1SVM (known as Support Vector Data Description (SVDD)) by Tax and Duin [8], and a Plane-based 1SVM (PSVM) by Scholkopf et al. [31], see Fig. 2; correspondingly, their hybrid models are referred to as a DBN-SVDD (DSVDD) and a DBN-PSVM (DPSVM), respectively. Note that the other types of 1SVMs that are described in Section 2 can also be applied in a similar way to our hybrid model.

3.1. Deep belief nets (DBNs)

DBNs are multi-layer generative models that learn one layer of features at a time from unlabelled data. Two significant properties of DBNs are their ability to perform non-linear dimensionality reduction on very large datasets and to learn high-dimensional manifolds from the data.

A DBN can be trained efficiently in a greedy layer-wise fashion by using a Restricted Boltzmann Machine (RBM). An RBM is a bipartite graph, with visible units v representing observations and hidden units h learning to represent features, i.e., it maps the input vectors v from an input space of dimension n to a feature space of dimension $d = |h|$, where $d < n$. Given a dataset $D_{m\times n}$ as input, an RBM maps it to $\boldsymbol{X}_{m\times d}$. RBMs are restricted in the sense that there are no connections between units at the same level, i.e., visible–visible or hidden–hidden connections, and the two layers of the graph are connected with symmetric weights $\mathbf{W}$ between pairs of hidden and visible units.

As in the original Boltzmann machine architecture, the joint distributions over hidden and visible vectors $p(\boldsymbol{v},\boldsymbol{h})$ are defined in terms of an energy function $E(\boldsymbol{v},\boldsymbol{h})$, as $p(\boldsymbol{v},\boldsymbol{h}) = \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{Z}$, where Z is a normalisation factor called the partition function and is calculated as $Z = \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$. The energy of the joint configuration is determined with respect to the values of the network parameters $\theta = (\mathbf{W},\mathbf{b},\mathbf{c})$, where $\mathbf{b}$ and $\mathbf{c}$ are biases to the hidden layer and visible layers, respectively. The hidden layer $\boldsymbol{h}$ is binary, and the hidden units are Bernoulli random variables, whereas the input units can be either binary or real-valued. Assuming that the input vectors are Gaussian random variables with variance σ, the energy function for this Gaussian–Bernoulli RBM can be obtained as:

$$E(\boldsymbol{v},\boldsymbol{h}) = \frac{1}{2}\sum_i (v_i - c_i)^2 - \sum_j b_j h_j - \sum_{i,j} w_{i,j} v_i h_j, \tag{1}$$

where v_i and h_j are the ith and jth units of the visible $\boldsymbol{v}$ and hidden $\boldsymbol{h}$ layers with the symmetric weight of $w_{i,j}$, respectively, and the corresponding biases c_i and b_j. Given a binary hidden unit h_j, since there is no connection between hidden units it is straightforward to calculate the conditional distribution $p(\boldsymbol{h}|\boldsymbol{v})$,

$$p(\boldsymbol{h}|\boldsymbol{v}) = \prod_j p(h_j|\boldsymbol{v}) = \prod_j Sigm\left(b_j + \sum_i w_{i,j} v_i\right), \tag{2}$$

and similarly since there is no connection between visible units, $p(\boldsymbol{v}|\boldsymbol{h})$ factorises to

$$p(\boldsymbol{v}|\boldsymbol{h}) = \prod_i p(v_i|\boldsymbol{h}) = \prod_i \mathcal{N}\left(c_i + \sum_j w_{i,j} h_j, \sigma\right), \tag{3}$$

where $Sigm(x) = \frac{1}{1+e^{-x}}$ is the logistic sigmoid function and $\mathcal{N}(\mu, \sigma)$ denotes a Gaussian distribution with mean μ and variance σ.

Training an RBM implies finding the values of the parameters θ such that the energy is minimised. One possible approach aims at maximising the log-likelihood of $\boldsymbol{v}$ that is estimated by its gradient with respect to the model parameters,

$$\frac{\partial \log\ p(\boldsymbol{v})}{\partial \theta} = E_{p(\boldsymbol{h}|\boldsymbol{v})}\left[\frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \theta}\right] - E_{p(\boldsymbol{v}|\boldsymbol{h})}\left[\frac{\partial E(\boldsymbol{h}, \boldsymbol{v})}{\partial \theta}\right]. \tag{4}$$

Since an exact calculation of the second term of the log-likelihood is intractable, the gradient can be estimated using a method known as Contrastive Divergence (CD) [43]. CD approximates the expectation through k iterations of Gibbs sampling (often $k=1$) to update the network weights, i.e.,

$$\frac{\partial \log\ p(\boldsymbol{v})}{\partial w_{i,j}} \approx \langle v_i h_j \rangle^0 - \langle v_i h_j \rangle^k, \tag{5}$$

where $\langle . \rangle^I$ represents the average value at contrastive divergence iteration I. After training an RBM, another RBM can be stacked on top of the first one, i.e., the inferred states of the hidden units, $\boldsymbol{X}_{m \times d}$, are used as the visible units for training the new RBM. The upper layers can be a Bernoulli–Bernoulli RBM, in which the main difference to the first layer lies in the binary visible units and the energy function [44]. Stacking RBMs enables one to model the significant dependencies between the hidden units of the earlier RBM. More specifically, multiple layers of RBMs can be stacked to produce different layers of non-linear feature detectors, which represent progressively more complex statistical structure in the data. In a stack of RBMs, the bottom-up recognition weights of the resulting DBN are used to initialise the weights of a multi-layer feed-forward neural network. This network can be employed as a tool for dimensionality reduction, or topped with a logistic regression layer and discriminatively fine-tuned by back-propagation for classification.

3.2. One-class SVM

For the second stage of our DBN-1SVM architecture, we avoid discriminative fine-tuning of the DBN and feed the output of the last hidden layer (bottleneck) $\boldsymbol{X}$ to a one-class SVM (see Fig. 1).

3.2.1. Support vector data description (SVDD)

SVDD essentially finds the smallest possible hypersphere around the majority of the training records, while leaving out some points to be excluded as anomalies. Given $\boldsymbol{X} = \{\boldsymbol{x}_l : l = 1, \ldots, m\}$ and $\boldsymbol{X} \in \mathbb{R}^d$, SVDD finds the most tightly fitting hypersphere that encompasses most of the data points, as shown in Fig. 2. Denoting the centre of the hypersphere by $\boldsymbol{a}$ and its radius by R, this hypersphere formulation involves solving the following quadratic programming optimisation problem,

$$\begin{aligned} \min_{\boldsymbol{a},\,R,\,\boldsymbol{\xi}} \quad & R^2 + \frac{1}{m\nu}\sum_{l}^{m} \xi_l \\ \text{s.t.} \quad & \| \phi(\boldsymbol{x}_l) - \boldsymbol{a} \|^2 \le R^2 + \xi_l, \\ & \forall l = 1, \ldots, m,\ \xi_l \ge 0. \end{aligned} \tag{6}$$

The term $\phi(.)$ is a non-linear function that maps data to a higher dimensional space $\mathbb{R}^d \to \mathbb{R}^q$, where $d < q$. The term ν is a user predefined regularisation parameter that governs the trade-off between the size of the hypersphere and the fraction of data points falling outside the hypersphere, i.e., the fraction of training examples that can be classified as anomalies. The terms ξ_l, $l = 1, \ldots, m$, are the slack variables that allow some of the data points to lie outside the hypersphere.

Let $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]^T$ and $0 \le \alpha_l \le \frac{1}{m\nu}$. The dual problem for the above primary problem in (6) is as follows

$$\begin{aligned} & \max_{\boldsymbol{\alpha}} \quad \sum_{l=1}^{m} \alpha_l (\boldsymbol{x}_l \cdot \boldsymbol{x}_l) - \sum_{l,t} \alpha_l \alpha_t (\boldsymbol{x}_l \cdot \boldsymbol{x}_t), \\ & \forall 1 \le l, t \le m, \\ & \text{s.t.} \quad 0 \le \alpha_l \le \frac{1}{m\nu}. \end{aligned} \tag{7}$$

Maximising this optimisation (7) gives a set of α_l. For the training samples $\boldsymbol{x}_l$ that satisfy the inequality $\| \phi(\boldsymbol{x}_l) - \boldsymbol{a} \|^2 < R^2 + \xi_l$, the corresponding Lagrange multiplier will be zero, i.e., $\alpha_l = 0$. For examples that satisfy the equality $\| \phi(\boldsymbol{x}_l) - \boldsymbol{a} \|^2 = R^2 + \xi_l$, the multiplier will become greater than zero, i.e., $\alpha > 0$. In summary,

$$\begin{aligned} & \| \phi(\boldsymbol{x}_l) - \boldsymbol{a} \|^2 < R^2 + \xi_l, \alpha = 0 \quad \text{inlier}, \\ & = R^2 + \xi_l, 0 < \alpha < \frac{1}{m\nu} \quad \text{border SVs}, \\ & > R^2 + \xi_l, \alpha = \frac{1}{m\nu} \quad \text{outlier}. \end{aligned}$$

The centre of the hypersphere is a linear combination of the training examples, i.e., $\boldsymbol{a} = \sum_i (\alpha_i x_i)$. Among the input vectors $\boldsymbol{x}_l$, only those with $\alpha_l > 0$ (the support vectors (SVs)) are required. After solving the above optimisation problem, a test instance $\boldsymbol{z}$ is detected as an anomaly if its distance to $\boldsymbol{a}$ is larger than the radius R,

$$\begin{aligned} & \| \phi(\boldsymbol{z}) - \boldsymbol{a} \|^2 > R^2, \\ & \text{where} \, \| \phi(\boldsymbol{z}) - \boldsymbol{a} \|^2 = (\boldsymbol{z} \cdot \boldsymbol{z}) - 2\sum \alpha_l (\boldsymbol{z} \cdot \boldsymbol{x}_l) + \sum_{l,t} \alpha_l \alpha_t (\boldsymbol{x}_l \cdot \boldsymbol{x}_t). \end{aligned}$$

3.2.2. Plane-based one-class support vector machine (PSVM)

An alternative to SVDD for the hybrid model is a plane-base 1SVM (PSVM) [31,45]. PSVM identifies anomalies in the feature space by finding a hyperplane that best separates the data from the origin. In other words, the decision function of PSVM returns $+1$ in a region where most of the data points occur (i.e., where the probability density is high), and returns -1 elsewhere. To separate the data from the origin, PSVM solves the following quadratic optimisation function:

$$\begin{aligned} \min_{\boldsymbol{s},\boldsymbol{\xi},\rho} \quad & \frac{1}{2}\| \boldsymbol{s} \|^2 + \frac{1}{m\nu}\sum_{l=1}^{m} \xi_i - \rho \\ \text{s.t.} \quad & (\boldsymbol{s} \cdot \boldsymbol{x}_l) \ge \rho - \xi_l, \\ & \forall l = 1, \ldots, m, \xi_l \ge 0. \end{aligned} \tag{8}$$

Since the non-zero slack variables ξ_l are penalised in the objective function, the PSVM estimates a decision function $f_{\boldsymbol{s},\rho}(\boldsymbol{x}) = \text{sgn}(\boldsymbol{s} \cdot \boldsymbol{\phi}(\boldsymbol{x}) - \rho)$ that maximises the distance of all the data points (in the feature space) from the hyperplane to the origin, parameterised by a weight vector $\boldsymbol{s}$ and an offset ρ.

By introducing the Lagrange multipliers and setting the primal variables $\boldsymbol{s}$, $\boldsymbol{\xi}$ and ρ equal to zero, the quadratic program can be derived as the dual of the primal program in Eq. (8):

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{lt} k(\boldsymbol{x}_l, \boldsymbol{x}_t) \\ \text{s.t.} \quad & 0 \le \alpha_l \le \frac{1}{m\nu}, \sum_{l} \alpha_l = 1. \end{aligned} \tag{9}$$

Using the Karush–Kuhn–Tucker optimality conditions (KKT conditions) the data vectors can be characterised in terms of whether they fall below, above, or on the hyperplane boundary in the feature space depending on the corresponding α_i values. Data vectors with positive α_i values are the support vectors. Further, for $0 < \alpha_i < 1/\nu n$, the data vectors fall on the hyperplane and hence ρ can be recovered using these vectors, vis-a-vis $\rho = \langle \boldsymbol{s}, \boldsymbol{\phi}(\boldsymbol{x}_i) \rangle = \sum_j \alpha_j k(\boldsymbol{x}_j, \boldsymbol{x}_i)$. Therefore, the decision function can now be written as

$$f_{\boldsymbol{s},\rho}(\boldsymbol{x}) = \mathrm{sgn}(\boldsymbol{s}\cdot\boldsymbol{\phi}(\boldsymbol{x})-\rho) = \mathrm{sgn}(\alpha_i k(\boldsymbol{x}_i,\boldsymbol{x})-\rho). \tag{10}$$

4. Evaluation and discussion

This section evaluates the performance of the proposed hybrid approaches through various experiments. More specifically the aims of these experiments are as follows:

- Evaluate the effect of the choice of kernel on the performance of 1SVM-based methods (4.2).
- Compare the performance of DBN-1SVMs against the standalone approaches (i.e., autoencoder, SVDD and PSVM), as well as PCA-SVDD and PCA-PSVM (a hybrid model consisting of Principle Component Analysis (PCA) and a 1SVM) on several high-dimensional datasets. Note that PCA is used as a reference feature extraction algorithm as it is one of the most prevalent approaches used in conjunction with SVMs[2] (4.3).
- Measure the effect of the network depth (number of hidden layers) on the performance of DBN-1SVMs (Section 4.4).
- Determine the effect of the number of training records on the performance of the hybrid model (Section 4.5).
- Assess the time- and memory-complexity of the hybrid models (Section 4.6).

4.1. Experimental methodology

Experimental setup: DBN-based algorithms are implemented in MATLAB with mini-batch learning, following the original scheme in [18,19]. Each RBM was first individually trained using one step Contrastive Divergence[3] ($k=1$) [43], to extract features, and used the features to train the next layer. Training a network implies finding parameters (network weight and bias) that minimise the reconstruction error between the inputs $\boldsymbol{x}$ and the reconstruction of $\boldsymbol{x}$ at the output $\overline{\boldsymbol{x}}$, $l(\boldsymbol{x},\overline{\boldsymbol{x}}) = \|\boldsymbol{x}-\overline{\boldsymbol{x}}\|^2$. Once the network was trained, the learned parameter values were used as initialisation values of a multilayer perceptron (MLP) with the same number of inputs and outputs. Then the network was fine-tuned by gradient descent to adjust the parameters. The whole process of pre-training and fine-tuning was performed in an unsupervised manner so far. When the autoencoder is used for anomaly detection, anomalies can be identified based on the history of the squared error between the inputs and outputs for the training records. Let e be the set of reconstruction error values of the $\boldsymbol{x}_i \in \mathbf{X}$, where $i=1,\ldots,m$. If the reconstruction error for a test sample is larger than the threshold $\tau=\mu(\mathrm{e})+3\sigma(\mathrm{e})$, where $\mu(\mathrm{e})$ and $\sigma(\mathrm{e})$ are the mean and standard deviation of the values in the set e, respectively, then the record is identified as anomalous, otherwise it is identified as normal.

When the network is used as a feature extractor, e.g., in the hybrid model, its bottom half (i.e., a DBN) is used to extract feature sets, which are then taken as input to train a one-class SVM in the usual way. For the one-class SVMs (PSVM and SVDD) LIBSVM [48] is employed, and for the PCA, the standard native MATLAB command `princomp` is used. For visualisation purposes, a tool called *improved Visual Assessment of cluster Tendency* (iVAT) [49] is applied to help visualise the possible number of clusters in, or the cluster tendency of, a set of objects. iVAT reorders the dissimilarity matrix of the given set of objects so that it can display any clusters as dark blocks along the diagonal of the image.

Datasets: The experiments are conducted on six real-life datasets and two synthetic ones. The real-life datasets are from the UCI Machine Learning Repository: (i) Forest Adult Gas Sensor Array Drift (Gas), Opportunity Activity Recognition (OAR), Daily and Sport Activity (DSA), and Human Activity Recognition using Smartphones (HAR), with dimensionalities of 54, 123, 128, 242, 315[4] and 561 features, respectively. One synthetic dataset is the “Banana” dataset, generated from a mixture of two banana shaped distributions, which are randomly moved in 100 dimensions. The other is a “Smiley” dataset, generated from a mixture of two compact Gaussians and an arc shaped distribution, resembling an smiley-face. The dataset contains 1000 dimensions and in any two dimensions the components of the face are randomly moved. All the records in each dataset are normalised between [0,1].

Although DBN-1SVMs are designed to overcome the challenges that arise for anomaly detection in high-dimensional and large datasets, the experiments are conducted on datasets with varying numbers of dimensions and records, to assess the effect of data size on their performance. In each experiment, 80% of records are randomly selected for training and 20% for testing, and then, respectively, mixed with 5% and 20% anomalous records, randomly drawn from $U(0,1)$ [50]. Note that training is performed in an unsupervised way, and labels are only used for testing.

The hyperparameters of DBN-based networks, learning rate (for pretraining 0.001–0.01, for fine tuning 0.1–1), number of epochs (for pretraining 5–10, for fine tuning 10–30), number of hidden units ($d \lll n$), are set based on the best performance on a validation set following [51]. The parameters of SVM based methods are selected via a grid-search, width ν $(0-1)$, and σ $(1-\infty)$ for SVDD [8], and γ $(2^{-15}, 2^{-13}, \ldots, 2^{3})$ and C $(2^{-5}, 2^{-3}, \ldots, 2^{15})$ for RBF kernel [52]. In the case of PCA the number of components κ is set such that 95% of the variance is retained.

Metrics: The Receiver Operating Characteristic (ROC) curve and the corresponding Area Under the Curve (AUC) are used to measure the accuracy of all the methods. The reported training/testing times are in seconds based on experiments that were implemented in MATLAB (2014b) and run on a machine with an Intel Core i7 CPU at 3.40 GHz, 8 GB RAM and RAM frequency of 799.0 MHz. The training time for the hybrid methods, DBN-1SVMs and PCA-1SVMs, includes the time to train the dimensionality reduction algorithm, extracting features and training a 1SVM with the features. The stated performance measures, AUC values and training/testing times, are the average of 1000 iterations for each experiment.

In the following reported results, the value given after the headings of DBN-based methods (i.e., DBN1, DBN2, D1x, D2x, AE1 and AE3) indicates the number of hidden layers for the corresponding experiment (e.g., DBN1 has one hidden layer, DBN2 has two hidden layers). D1x and D2x are used as an abbreviation for referring to the hybrid methods, e.g., DBN1-SVDD is presented as D1SVDD. Note that all the autoencoders herein include an odd number of hidden layers, while the hybrid methods may have any number of hidden layers, for example, DBN2 corresponds to the encoder part of AE3, see Fig. 1.

Statistical analysis: Statistical analysis is conducted to identify statistical significance among the performance results obtained

[2] Kernel-PCA (KPCA) is another well-known nonlinear approach that can also be used as a dimensionality reduction tool. However, similar to other kernel-machines, KPCA suffers from high computational complexity ($O(N^3)$), i.e., it starts by projecting data to a higher dimensional space. Moreover, the main difficulty in KPCA is the choice of an appropriate kernel and the corresponding parameters, where choosing an inappropriate kernel may lead to an increase in dimensionality [46].

[3] In this paper we follow the original scheme in [18,19,43] to train an RBM, but other approaches like [47] can also be applied.

[4] DSA is a large dataset comprising the time series measurements from 45 wearable sensors for 19 activities. We select a portion of the time series for each of the first 7 activities, yielding a total of 315 concatenated time series features.

from the various methods. Non-parametric tests are chosen in this paper, according to the recommendation made in [53–56], where their use in the field of machine learning is strongly recommended. These tests are conducted since the initial conditions that guarantee the reliability of the parametric tests may not be satisfied, causing the statistical analysis to lose credibility with these parametric tests [53].

Throughout the experiments several non-parametric tests are considered. The Wilcoxon signed-rank test [57] is used to perform pairwise comparisons between different methods. The Iman–Davenport test [58] is used to detect statistical differences among a group of studied methods. If a significant difference was detected then a post hoc test, the Shaffer test [59], is performed to identify which methods are distinctive in an all-by-all comparison. The post hoc test is conducted to examine if a hypothesis of a comparison of means could be rejected at a specified significance level α. Therefore, the p-value associated with each comparison is computed, representing the lowest level of significance of a hypothesis that results in a rejection. This value allows one to identify if two algorithms have significantly different performance and to what extent. For all the comparisons in this study the significance level α is set to 0.1.

Table 1
Performance results. Training and testing times are in seconds, except for testing time rows marked with a *, which are in milliseconds. The results of the underscored methods are only considered in Section 4.2 and not used in the subsequent tests in the latter sections. The AUC$_{std}$ value for each method represents the standard deviation of the corresponding AUC over 1000 iterations.

Kernel	Hybrid	Method	Metric	Datasets								Avg.
				Forest	Banana	Adult	GAS	OAR	DSA	Smiley	HAR	
Linear		PSVM	AUC	0.83	0.77	0.81	0.89	0.87	0.78	0.75	0.80	0.8125
			AUC$_{std}$	± 0.07	± 0.11	± 0.08	± 0.08	± 0.12	± 0.06	± 0.09	± 0.15	± 0.0887
			Train	**0.0046**	**0.0121**	**0.0073**	0.0857	0.0722	**0.0156**	**0.0605**	**0.0791**	**0.0421**
			Test	0.0018	0.0029	0.0036	0.0063	0.0067	0.0085	0.0086	0.0082	0.0058
		SVDD	AUC	0.83	0.78	0.81	0.89	0.87	0.79	0.74	0.80	0.8137
			AUC$_{std}$	± 0.05	± 0.13	± 0.07	± 0.03	± 0.09	± 0.07	± 0.13	± 0.11	± 0.0850
			Train	0.0743	0.2505	0.1880	0.3175	0.2764	0.8314	1.9246	1.6288	0.6864
			Test	0.0019	0.0012	0.0015	0.0027	0.0004	0.0013	0.0025	0.0016	0.0016
RBF		PSVM	AUC	0.97	0.92	0.86	0.91	0.90	0.85	0.79	0.87	0.8838
			AUC$_{std}$	± 0.02	± 0.03	± 0.05	± 0.04	± 0.05	± 0.03	± 0.08	± 0.09	± 0.0488
			Train	0.0254	0.0464	0.0611	**0.0606**	0.0475	0.1516	1.0071	0.3548	0.2193
			Test	0.0091	0.0085	0.0286	0.0132	0.0090	0.0275	0.0976	0.0763	0.0337
		SVDD	AUC	0.97	0.92	0.87	0.91	0.91	0.84	0.78	0.88	0.8850
			AUC$_{std}$	± 0.02	± 0.04	± 0.02	± 0.04	± 0.06	± 0.05	± 0.05	± 0.07	± 0.0350
			Train	0.0192	0.0201	0.0423	0.0411	0.0165	0.0664	0.5422	0.2562	0.1255
			Test	0.0085	0.0986	0.0186	0.0123	0.0136	0.0243	0.0885	0.0828	0.0434
		AE3	AUC	**0.99**	0.97	**0.99**	**0.98**	**0.98**	**0.98**	0.96	**0.99**	0.9800
			AUC$_{std}$	± 0.00	± 0.01	± 0.00	± 0.00	± 0.00	± 0.00	± 0.01	± 0.00	± 0.0025
			Train	0.3217	0.518	0.564	0.99	0.67	0.63	1.7561	1.02	0.8093
			Test	0.0021	0.0035	0.0033	0.0029	0.0035	0.0041	0.0076	0.0054	0.0041
Linear	PCA	PSVM	AUC	0.84	0.77	0.82	0.91	0.88	0.78	0.76	0.80	0.8200
			AUC$_{std}$	± 0.07	± 0.09	± 0.07	± 0.06	± 0.08	± 0.03	± 0.06	± 0.11	± 0.0731
			Train	0.0283	0.0507	0.0661	0.1086	0.1438	0.2191	0.4217	0.2913	0.1662
			Test	0.0010	0.0089	0.0017	0.002	0.0024	0.0038	0.0053	0.0037	0.0036
		SVDD	AUC	0.84	0.78	0.82	0.91	0.88	0.79	0.77	0.80	0.8238
			AUC$_{std}$	± 0.06	± 0.08	± 0.08	± 0.07	± 0.04	± 0.05	± 0.08	± 0.10	± 0.0700
			Train	0.1146	0.0338	0.1643	0.1451	0.1717	0.4317	0.4911	0.5176	0.2587
			Test*	3.2717	0.0345	0.5154	0.6542	0.7879	1.2140	1.8087	2.0619	1.2936
RBF	PCA	PSVM	AUC	0.96	0.94	0.88	0.95	0.96	0.91	0.79	0.92	0.9137
			AUC$_{std}$	± 0.02	± 0.03	± 0.06	± 0.02	± 0.03	± 0.04	± 0.05	± 0.06	± 0.0388
			Train	0.1034	0.0228	0.1320	0.1647	**0.0452**	0.3252	0.4829	0.4944	0.2213
			Test	0.0176	0.0011	0.0240	0.0338	0.0055	0.0554	0.0232	0.0713	0.0290
		SVDD	AUC	0.96	0.94	0.89	0.95	0.96	0.91	0.80	0.92	0.9162
			AUC$_{std}$	± 0.01	± 0.04	± 0.05	± 0.02	± 0.01	± 0.03	± 0.04	± 0.04	± 0.0300
			Train	0.1156	0.1121	0.0654	0.0784	0.1352	0.2126	0.2916	0.3490	0.2436
			Test	0.0088	0.0031	0.0018	0.0062	0.0063	0.0206	0.0308	0.0256	0.0129
Linear	DBN2	PSVM	AUC	**0.99**	**0.99**	0.98	**0.98**	**0.98**	**0.98**	0.98	**0.99**	**0.9838**
			AUC$_{std}$	± 0.00	± 0.00	± 0.01	± 0.00	± 0.00	± 0.01	± 0.01	± 0.00	± 0.0013
			Train	0.0768	0.0815	0.0898	0.0969	0.0803	0.1495	0.3015	0.2416	0.1289
			Test*	**0.0082**	**0.0095**	**0.0091**	**0.0082**	**0.0167**	**0.0092**	**0.0111**	**0.0096**	**0.0102**
		SVDD	AUC	**0.99**	**0.99**	**0.99**	0.97	**0.98**	**0.98**	0.98	**0.99**	**0.9838**
			AUC$_{std}$	± 0.00	± 0.00	± 0.00	± 0.01	± 0.00	± 0.01	± 0.01	± 0.00	± 0.0025
			Train	0.0768	0.0815	0.0898	0.0969	0.0803	0.1495	0.3225	0.2416	0.1289
			Test*	0.018	0.0397	0.145	0.5597	0.5076	0.4738	1.3518	1.1496	0.5307
RBF	DBN2	PSVM	AUC	0.98	**0.99**	**0.99**	0.96	0.97	**0.98**	**0.99**	**0.99**	0.9813
			AUC$_{std}$	± 0.00	± 0.00	± 0.00	± 0.01	± 0.01	± 0.00	± 0.00	± 0.00	± 0.0025
			Train	0.0903	0.1174	0.1256	0.1048	0.1379	0.1715	0.3265	0.2404	0.1560
			Test	0.007	0.0105	0.0086	0.0067	0.0032	0.0068	0.0153	0.0088	0.0084
		SVDD	AUC	**0.99**	**0.99**	**0.99**	0.97	**0.98**	**0.98**	0.98	**0.99**	**0.9838**
			AUC$_{std}$	± 0.00	± 0.00	± 0.00	± 0.01	± 0.00	± 0.00	± 0.01	± 0.00	± 0.0025
			Train	0.0909	0.0974	0.1212	0.1075	0.1566	0.1881	0.3642	0.3186	0.1749
			Test*	0.1213	0.1871	0.1434	0.1831	0.1260	0.2581	0.2253	0.1628	0.1759

4.2. Choice of kernel

In Section 3 two types of one-class SVMs, hypersphere and hyperplane SVM, were presented for combination with a DBN. Given that each of the methods can be used in conjunction with different kernels, their performance for two common kernels, linear and RBF, are evaluated. Table 1 shows the performance results and their average over all the datasets. For further information, the standard deviation of the AUC values are also included as AUC$_{std}$ in the table. The *best* case, i.e., the *highest* AUC and *lowest* training/testing time, for each dataset is stressed through *boldface*.

Before taking the study further, to improve the clarity in the reporting results only the best pair of kernel and 1SVM methods is taken. The Wilcoxon test is conducted to perform a pair-wise comparison among the performance values of PSVM and SVDD when used with linear and RBF kernels. Table 2 demonstrates the output of this comparison on all the hybrid and stand-alone 1SVM results from Table 1. The returned *p*-values in either case fail to reject the null hypothesis for the accuracy measure with a level of significance of $\alpha = 0.1$, i.e., the accuracy of PSVM and SVDD are comparable for both kernels. Similar results are obtained for the training time of the 1SVM methods used with the RBF kernel, whereas in case of the linear kernel, PSVM outperforms SVDD, i.e., the sum of ranks for PSVM is less than the sum of ranks for SVDD, meaning that the PSVM performance was generally better. A discrepancy is observed in the testing time, where the linear kernel PSVM outperforms SVDD, while for the RBF kernel SVDD outperforms PSVM. Consequently, hereafter only the results for PSVM with the linear kernel and SVDD with the RBF kernel from Table 1 are considered in this study. Note that by 1SVM we refer to these two methods with the above combination/setup.

4.3. Influence of dimensionality reduction

While the main intention behind applying a dimensionality reduction method to an SVM is to alleviate the computational complexity, it is also interesting to investigate its effect on accuracy. Table 1 compares the differences in the performance of the hybrid and stand alone methods on the experimental frame work detailed above. As expected, the results obtained using the 1SVM approaches exhibit poor performance for high-dimensional datasets. In contrast, the results from the hybrid 1SVMs suggest that a feature extraction method such as PCA or DBN is able to enhance the accuracy of the baseline techniques. Between the two, DBN achieved the best accuracy. On average the PCA-based 1SVMs exhibit an increase of about 1% in their AUC, while SVDD and PSVM experience a 10% and 16% increase in AUC, respectively, when combined with a DBN. In this way, the DBN-1SVM can achieve a comparable AUC to AE3. This improvement is due to the fact that DBNs are better in characterising highly non-linear functions with many variations [17]. In contrast, the objective of PCA is to learn a linear manifold that is closest to the distribution of the training samples, i.e., characterising a lower-dimensional region in the input space near where the training records have a higher density. However, for complex real world domains, data manifolds are likely to be strongly non-linear.

Table 2
Wilcoxon test to compare the performance of the PSVM and SVDD based methods with respect to the choice of kernel. R$^+$ corresponds to the sum of the ranks for the method on the left and R$^-$ for the right.

PSVM vs. SVDD	Accuracy			Training time			Test time		
	R$^+$	R$^-$	*p*-value	R$^+$	R$^-$	*p*-value	R$^+$	R$^-$	*p*-value
Linear	2	7	0.2891	48	136	**0.0004**	62	238	**0.0119**
RBF	3	7	0.1826	193	107	0.2192	203	97	**0.0024**

In the hybrid models, a similar improvement can also be observed for the testing time, but not in the training time. Compared with the standalone methods, the results show a substantial reduction in the testing time of the hybrid models for all the datasets, i.e., the average testing time of SVDD and PSVM are reduced by a factor of approximately 2 when aggregated with PCA, and by 150 and 5000 when aggregated with DBN. A trade-off for the hybrid models is their training time, which shows an increase against SVDD and PSVM, i.e., the training time of the hybrid SVDD methods are increased by half, and for the hybrid PSVM methods are tripled.

In order to statistically compare the performance of the studied anomaly detection methods and find the performance relationship among them, a multiple comparison test is conducted on the reported results of Table 1. For ease of interpreting these results, Fig. 3 graphically demonstrates the average rankings of the three metrics, AUC, training and testing time, obtained using Friedman's test [60].

Under all the measures, a statistical study conducted using the Iman–Davenport test detects significant differences between the algorithms and rejects the null hypothesis of equivalence between the methods. The returned *p*-values — i.e., 0.69e^{-19} for AUC, 0.12 e^{-18} for training time, and 0.68e^{-19} for testing time — are significantly lower than the significance level α-value of 0.1. The identified differences are then analysed with a Shaffer post hoc test, shown in Tables 3–5. In these tables, "+" symbol indicates that the method in the row is statistically *better* than the one in the column, i.e., in the case of AUC it implies a *higher* accuracy and in the case of the time it implies a *lower* running time, whereas "−" implies the contrary; "=" indicates that the two compared methods have no significant differences.

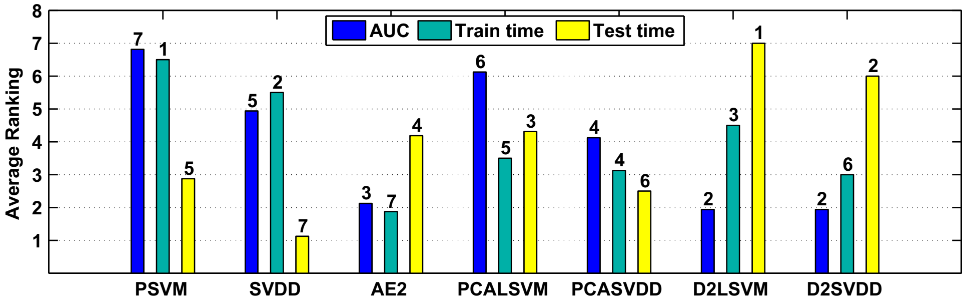

Fig. 3. Comparison of rankings of anomaly detection methods for 3 metrics. The bars represent average rankings based on the Friedman test, and the number on the top of the bars indicates the ranking of the algorithm, from the best (1) to worst (7) for each given measure, and if a tie occurs then the best mean result is taken. The ranking is determined for all datasets and finally an average is calculated as the mean of all rankings.

Observing the results from Tables 3 and 5, it can be noted that the DBN-based methods (AE3, D2SVDD and D2PSVM) clearly outperform their other counterparts in terms of AUC and testing time. It can also be inferred that the DBN-based methods generate more robust results, having negligible standard deviations of the AUC values from Table 1. Although DBN-based methods are not the leading approaches in terms of training time, as shown in Table 4, it can be concluded that DBN is a powerful tool as a dimensionality reduction method, when combined with 1SVMs. Moreover, it overcomes the underlying scalability issues of 1SVMs, and improves their performance.

In addition, it can be seen from the table that the performance of the D2SVDD and D2PSVM approaches are not significantly different from AE3. To assess the significance of the results, the Wilcoxon test is applied to conduct a multiple pairwise comparison. Table 6 shows the p-values obtained from this test, which indicate that the AUC values of the three methods are statistically similar. However, both hybrid approaches (D2PSVM and D2SVDD) outperform AE3 in terms of training and testing time, at the significance level of $\alpha = 0.1$, with D2PSVM being in the lead.

From the reported results in Tables 1, 3 and 5, the positive synergy between the DBN and the 1SVM methods can be observed, offering an outstanding anomaly detection method for high-dimensional datasets. The next section studies the influence of DBN depth on the performance of these hybrid methods.

4.4. Influence of number of hidden layers

Increasing the number of hidden layers (i.e., the depth of the representation) has shown promising results in tackling data complexity and the curse of dimensionality issues [61,17]. However, that is not the case for linear algorithms such as PCA. Stacking up PCAs with the intention of forming a deeper architecture and obtaining a more abstract representation is ineffective. The composition of linear operations yields another linear operation [61]. Thus, only the impact of network depth on DBN-based methods is studied. Table 7 summarises the performance results of AEs and DBN-1SVMs with various numbers of hidden layers, e.g., AE1 indicates an AE with one hidden layer. Analogous to the earlier results, on average the hybrid methods present a higher AUC value and lower training/testing time.

Comparing the DBN1-based results with deeper networks in Tables 1 and 7, the experiments indicate that increasing the number of hidden layers enhances anomaly detection, in both the baseline AE and the hybrid methods. However, appropriate statistical analysis is required to support this claim. For this purpose, the Wilcoxon signed-rank test is applied to highlight the significance of the differences. Table 8 shows the returned p-values. In concordance with previous studies [16], the experimental results suggest that the deeper the network, the greater the accuracy and training/testing time. The Wilcoxon test shows that deeper networks increase the accuracy of shallower ones. This improvement is more significant when

Table 3
Shaffer test to compare AUC values.

Method	PSVM	SVDD	AE3	PCAPSVM	PCASVDD	D2PSVM	D2SVDD
PSVM	x	$-(0.08)$	$-(0.14e^{-4})$	$=(0.52)$	$-(0.01)$	$-(0.63e^{-5})$	$-(0.63e^{-5})$
SVDD	$+(0.08)$	x	$-(0.92e^{-2})$	$=(0.27)$	$=(0.45)$	$-(0.55e^{-2})$	$-(0.55e^{-2})$
AE3	$+(0.14e^{-4})$	$+(0.92e^{-2})$	x	$+(0.21e^{-3})$	$+(0.06)$	$=(0.86)$	$=(0.86)$
PCAPSVM	$=(0.52)$	$=(0.27)$	$-(0.21e^{-3})$	x	$-(0.06)$	$-(0.11e^{-3})$	$-(0.11e^{-3})$
PCASVDD	$+(0.01)$	$=(0.54)$	$-(0.06)$	$+(0.06)$	x	$-(0.04)$	$-(0.04)$
D2PSVM	$+(0.63e^{-5})$	$+(0.55e^{-2})$	$=(0.86)$	$+(0.11e^{-3})$	$+(0.04)$	x	$=(1.0)$
D2SVDD	$+(0.63e^{-5})$	$+(0.55e^{-2})$	$=(0.86)$	$+(0.11e^{-3})$	$+(0.04)$	$=(1.0)$	x

Table 4
Shaffer test to compare training times.

Method	PSVM	SVDD	AE3	PCAPSVM	PCASVDD	D2PSVM	D2SVDD
PSVM	x	$=(0.35)$	$+(0.35e^{-6})$	$+(0.54e^{-2})$	$+(0.17e^{-2})$	$+(0.06)$	$+(0.11e^{-2})$
SVDD	$=(0.35)$	x	$+(0.31e^{-4})$	$+(0.06)$	$+(0.03)$	$=(0.35)$	$+(0.02)$
AE3	$-(0.35e^{-6})$	$-(0.31e^{-4})$	x	$-(0.02)$	$-(0.05)$	$-(0.12e^{-2})$	$-(0.06)$
PCAPSVM	$-(0.54e^{-2})$	$-(0.06)$	$+(0.02)$	x	$=(0.72)$	$=(0.35)$	$=(0.64)$
PCASVDD	$-(0.17e^{-2})$	$-(0.03)$	$+(0.05)$	$=(0.72)$	x	$=(0.20)$	$=(0.91)$
D2PSVM	$-(0.06)$	$=(0.35)$	$+(0.12e^{-2})$	$=(0.35)$	$=(0.20)$	x	$=(0.16)$
D2SVDD	$-(0.11e^{-2})$	$-(0.02)$	$+(0.06)$	$=(0.64)$	$=(0.91)$	$=(0.16)$	x

Table 5
Shaffer test to compare test times.

Method	PSVM	SVDD	AE3	PCAPSVM	PCASVDD	D2PSVM	D2SVDD
PSVM	x	$-(0.08)$	$=(0.56)$	$=(0.13)$	$=(0.72)$	$-(0.21e^{-3})$	$-(0.54e^{-2})$
SVDD	$+(0.08)$	x	$-(0.02)$	$-(0.11e^{-2})$	$=(0.16)$	$-(0.53e^{-7})$	$-(0.63e^{-5})$
AE3	$=(0.56)$	$+(0.02)$	x	$=(0.35)$	$=(0.35)$	$-(0.17e^{-2})$	$-(0.03)$
PCAPSVM	$=(0.13)$	$+(0.11e^{-2})$	$=(0.35)$	x	$-(0.06)$	$-(0.03)$	$=(0.20)$
PCASVDD	$=(0.72)$	$=(0.16)$	$=(0.35)$	$+(0.06)$	x	$-(0.51e^{-4})$	$-(0.17e^{-2})$
D2PSVM	$+(0.21e^{-3})$	$+(0.53e^{-7})$	$+(0.17e^{-2})$	$+(0.03)$	$+(0.51e^{-4})$	x	$=(0.35)$
D2SVDD	$+(0.54e^{-2})$	$+(0.63e^{-5})$	$+(0.03)$	$=(0.20)$	$+(0.17e^{-2})$	$=(0.35)$	x

comparing DBN1-based networks with DBN2-based (i.e., D2x and AE3) networks, and the comparison rejects the null hypothesis with a level of significance of $\alpha = 0.1$, whereas in the case of DBN3 (i.e., D3x and AE5) and DBN2 methods the returned *p*-values are not statistically significant. This statement also holds for the training measures, increasing the network's depth increases the training time. However, unlike the accuracy comparison, the null hypothesis is only rejected in comparisons among the deeper networks, the DBN2 and DBN3 methods. This indicates that in comparisons among the deeper networks, the accuracy improvement is not statistically significant, while the efficiency of the deeper networks has decreased.

Based on these results, it can be concluded that deeper networks are able to infer more abstract features. Accordingly, more abstract features are generally more invariant to underlying variations in the training set. However, it should be considered that increasing the depth of the network beyond a certain point may only increase the complexity and execution time of the network.

Overall, the DBN2-based methods deliver the best performance, both in terms of accuracy and efficiency. Combining DBN2 and 1SVMs, therefore, results in a more robust anomaly detection technique, while maintaining the training/testing time as low as possible. Recall that PSVM obtains comparable AUC as SVDD, when combined with DBN2. Hence, the greater computational efficiency of D2PSVM (see Table 6) makes it a more practical anomaly detection technique.

Table 6
Wilcoxon test to compare the performance of DBN-based methods regarding *p*-values of Wilcoxon test. R^+ corresponds to the sum of the ranks for the method on the left and R^- for the right.

Method	Accuracy			Train time			Test time		
	R^+	R^-	*p*-value	R^+	R^-	*p*-value	R^+	R^-	*p*-value
D2PSVM vs. AE3	5	1	0.5	0	35	**0.0078**	0	36	**0.0078**
D2SVDD vs. AE3	5	1	0.5	0	36	**0.0078**	0	36	**0.0078**
D2PSVM vs. D2SVDD	1	1	1	0	36	**0.0078**	0	36	**0.0078**

The reason behind this phenomenon can be explored in Fig. 4, which demonstrates the benefits of using DBN features for anomaly detection on two datasets, Banana and HAR (due to the shortage of space the images of the other datasets are not shown). The interpretation of the subfigures, from the left, is as follows: the iVAT image of the raw datasets, the iVAT image of the reduced datasets with one hidden layer, and with two hidden layers. As can be seen in the image of the raw datasets, normal records (first 760 records) appear in dense blocks, while the anomalous records (last 40 records) are shown in gray shadow (since they are distributed across the dataset). When the data are projected to a lower dimensional space with DBN, a clearer separation appears between the normal records and anomalies. The impact of the depth of the network is reflected in the improved clarity and contrast in the block structure. Projecting data with deeper networks (e.g., DBN2) provides sufficient separability among the normal and anomalous records that to enable the use of basic kernels such as linear, rather than more complex ones such as RBF.

Table 8
Wilcoxon tests to determine the influence of the number of hidden layers on the performance of DBN-based methods. R^+ corresponds to the sum of the ranks for the method on the left and R^- for the right.

Dataset	Accuracy			Train time			Test time		
	R^+	R^-	*p*-value	R^+	R^-	*p*-value	R^+	R^-	*p*-value
AE3 vs. AE1	33	0	**0.0078**	28	8	0.1484	23.5	4.5	**0.0391**
AE3 vs. AE5	0	5	0.2500	1	35	**0.0156**	0	36	**0.0078**
D2PSVM vs. D1PSVM	28	0	**0.0156**	35	1	0.7422	19	17	0.8438
D2PSVM vs. D3PSVM	0	3	0.2500	0	36	**0.0078**	0	36	**0.0078**
D2SVDD vs. D1SVDD	15	0	**0.0625**	26	8	0.1953	36	0	**0.0078**
D2SVDD vs. D3SVDD	0	3	0.5000	0	36	**0.0078**	0	36	**0.0078**

Table 7
Performance results of deep and shallow DBN-based methods. Training and testing times are in seconds, except for testing time rows marked with a *, which are in milliseconds.

Method	Metric	Datasets								Avg.
		Forest	Banana	Adult	GAS	OAR	DSA	Smiley	HAR	
AE1	AUC	0.97	0.93	0.98	0.96	0.94	0.97	0.91	0.98	0.9550
	Train	0.104	0.411	0.267	0.352	0.433	0.383	0.989	1.7320	0.5839
	Test	0.002	0.0033	0.003	0.0031	0.0032	0.0039	0.0068	0.0047	0.0037
AE5	AUC	**0.99**	0.98	**0.99**	**0.99**	**0.98**	**0.98**	**0.99**	**0.99**	0.9850
	Train	0.5162	0.6802	0.7843	2.015	0.8468	0.63	2.003	2.0216	1.1871
	Test	0.0068	0.0043	0.0102	0.0056	0.0051	0.0058	0.0098	0.0081	0.0070
D1PSVM	AUC	0.98	0.93	0.98	0.96	0.95	0.97	0.94	0.98	**0.9612**
	Train	**0.0304**	**0.0649**	0.1042	**0.0813**	0.0823	**0.1502**	0.3632	0.3811	**0.1572**
	Test *	0.0100	0.0120	0.0100	**0.0087**	**0.0093**	**0.0088**	**0.0089**	**0.0081**	**0.0195**
D3PSVM	AUC	**0.99**	**0.99**	**0.99**	**0.99**	**0.98**	0.98	**0.99**	**0.99**	**0.9863**
	Train	0.0916	0.0932	**0.0963**	0.1033	**0.0818**	0.1546	**0.3319**	0.3019	0.1568
	Test *	**0.0096**	**0.0100**	**0.0093**	0.0094	0.0194	0.0104	0.0213	0.0115	**0.0126**
D1SVDD	AUC	0.98	0.95	**0.99**	0.97	0.95	0.97	0.94	**0.99**	**0.9675**
	Train	0.0827	0.0932	0.1218	0.0992	0.1498	0.1832	0.3758	**0.2886**	0.1743
	Test *	0.1011	0.128	0.14	0.104	0.106	0.218	0.1850	0.1430	0.1406
D3SVDD	AUC	**0.99**	**0.99**	**0.99**	**0.99**	**0.98**	**0.98**	**0.99**	**0.99**	**0.9863**
	Train	0.0988	0.1048	0.1236	0.1129	0.1618	0.1925	0.4211	0.3507	0.1958
	Test *	0.1381	0.2051	0.1864	0.2127	0.1756	0.2762	0.2873	0.1928	0.2093

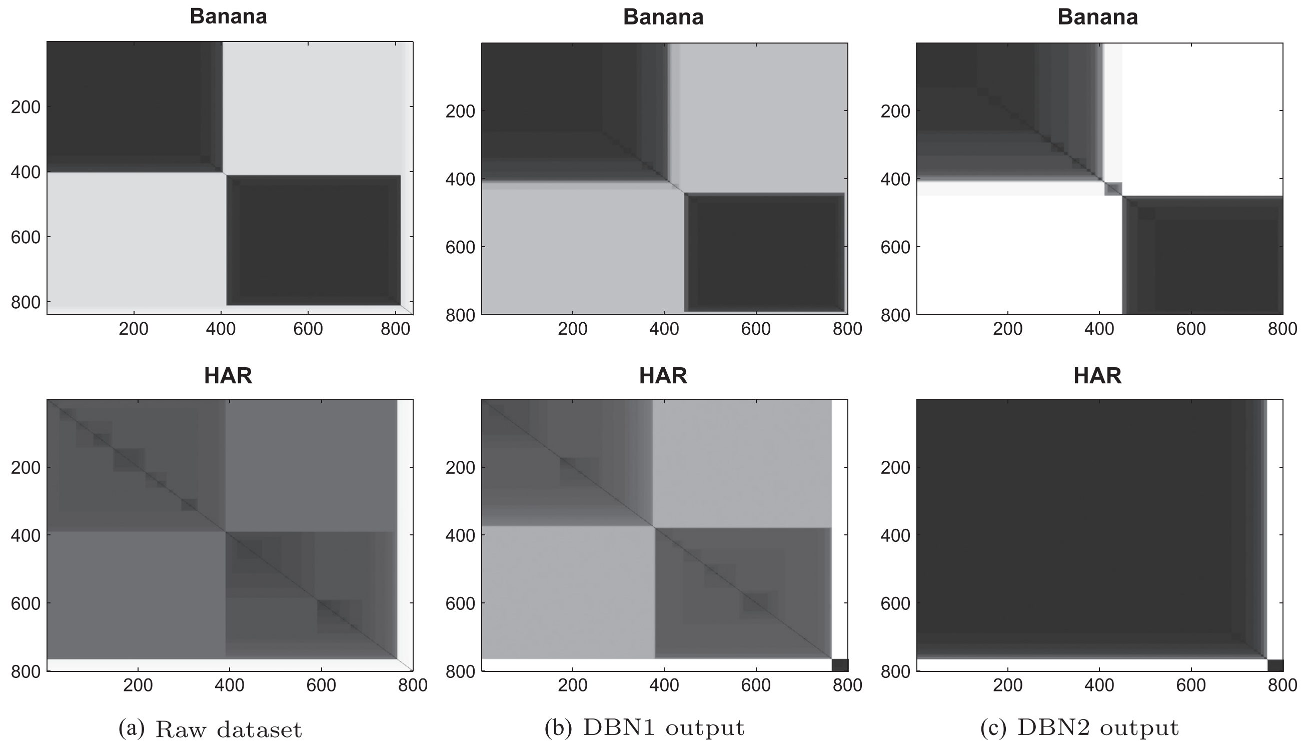

Fig. 4. Demonstration of the impact of the number of hidden layers on the separability of normal and anomalous records. Each image is the output of the iVAT method for visualising cluster tendency.

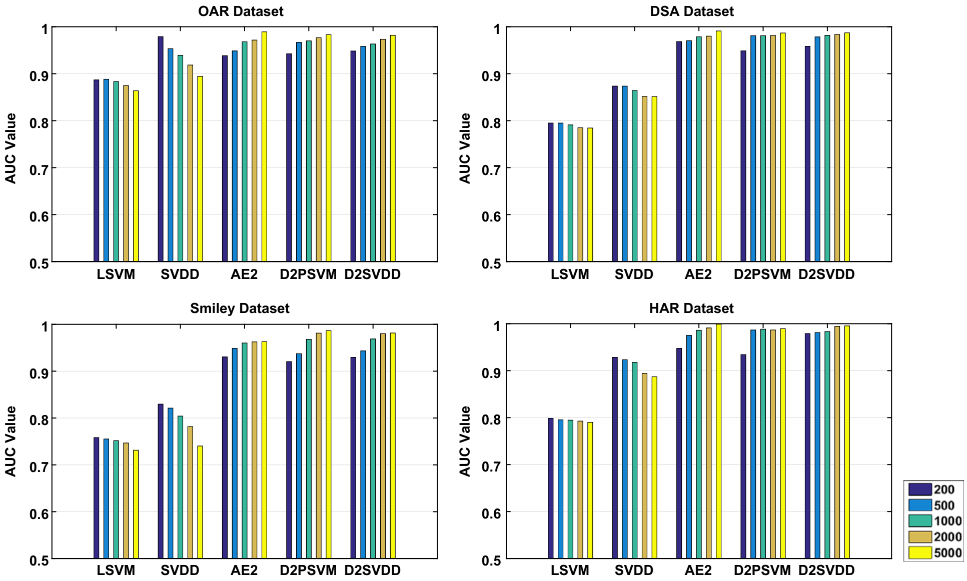

Fig. 5. Comparison of the accuracy of anomaly detection methods as the number of training records is varied.

4.5. Influence of number of training records

Unlike deep networks, SVMs are known to be less effective for large-scale training on high-dimensional data [14,4], i.e., their performance decreases as the number of records in the dataset increases. To study this concern and also examine the performance of our hybrid methods, PSVM, SVDD, AE3, D2PSVM and D2SVDD are trained on the four datasets with the highest dimensionality (HAR, Smiley, DSA and OAR) while varying the number of records (from 200 to 5000). As shown in Fig. 5, with 1000 or more training records, DBN-based methods outperform PSVM and SVDD. When the number of records is particularly small (less than 500), SVDD can perform slightly better than DBN-based methods. More specifically, the accuracy of the DBN-based methods generally increases as the number of records grows, whereas the 1SVM methods show the reverse behaviour. This suggests that the 1SVMs tend to learn models with higher variance for large datasets, i.e., they overfit the data.

4.6. Efficiency and scalability

When it comes to large-scale training, in addition to robustness, it is advantageous for an anomaly detection algorithm to be efficient in terms of time- and memory-complexity. Table 9 compares the complexity of PSVM, SVDD, DBN, DPSVM and DSVDD. As can be seen from the table, SVDD and DSVDD with the RBF kernel [13] have much higher (quadratic) training- and memory-complexity compared to the other techniques. Although the complexity of DPSVM may not be as efficient as PSVM, note that it still grows linearly.

To explore the efficiency of the DPSVM in practice, we compare it with an AE, an effective method with linear complexity, on three datasets, Adult, Forest each with over 40,000 records and Smiley dataset with 1.1 million records (see Fig. 6). This figure suggests that although training and testing time grow linearly for the two techniques, the training/testing time of the AE grows at a much faster rate. The AUC values for this experiment are not included, since they were more or less consistent.

Table 9
Comparison of time- and memory-complexities.

Technique	PSVM	SVDD	AE	DSVDD	DPSVM
Training	nm	$O(nm^2)$	$O(nmd)$	$O(nmd+dm^2)$	$O(nmd+dm)$
Testing	$n+m$	$O(nm)$	$O(nmd)$	$O(nmd+dm^2)$	$O(nmd+d+m)$
Memory	nm	$O(nm^2)$	$O(nq)$	$O(nq+dm^2)$	$O(nq+dm)$

In summary, from the aforementioned analysis we can conclude that the proposed hybrid methods are more suitable and robust than the standalone techniques for large-scale and high-dimensional anomaly detection. Recall that the efficiency and accuracy of D2PSVM make it the most effective approach, even when compared with the state-of-the-art AE. Fig. 7 demonstrates that the proposed hybrid methods have the best performance, the highest the AUC and lowest training/testing time, among all the studied methods, with D2PSVM outperforming D2SVDD in computation time.

5. Conclusion

In this paper, we proposed an unsupervised anomaly detection technique for high-dimensional large-scale unlabelled datasets. The technique is a combination of a DBN and one-class SVM. The

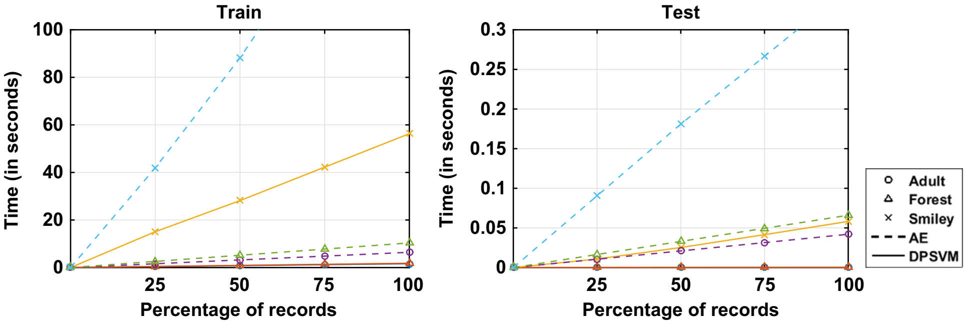

Fig. 6. Comparison of the training and testing time of AE and DPSVM on large datasets. In this experiment the Adult and Forest datasets include 40,000 records and the Smiley dataset includes 1.1 million records.

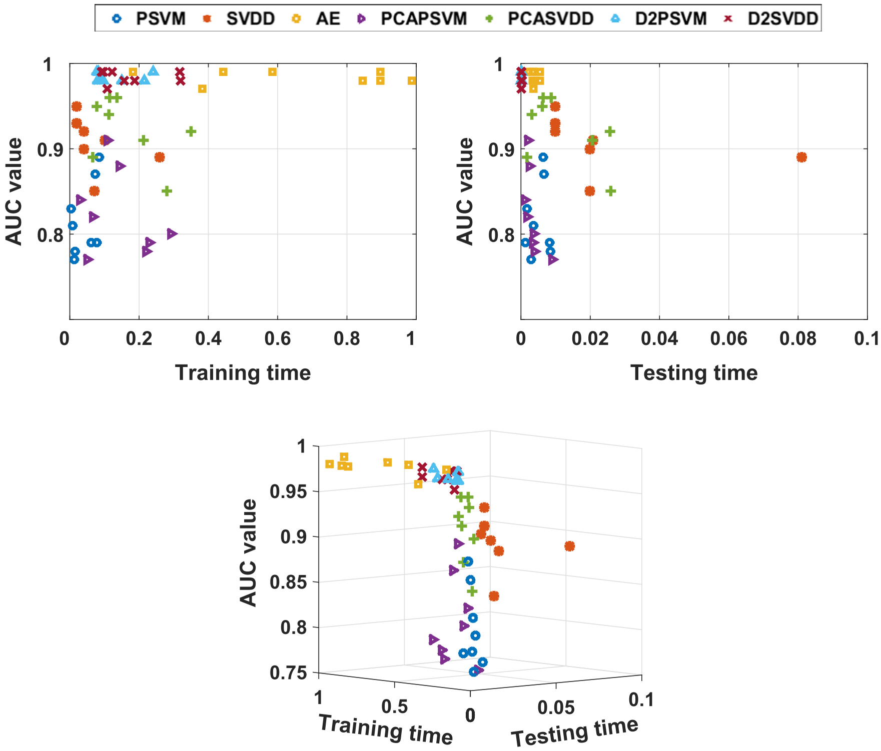

Fig. 7. Comparing the performance results of the proposed and studied anomaly detection techniques.

DBN is trained as a dimensionality reduction algorithm, generating a non-linear manifold and transforming the data into a lower dimensional set of features. The derived features from the training samples are taken as input to train the one-class SVM. Subsequently, the generated hybrid model from the two algorithms is used for testing.

Coupling a DBN with a 1SVM is advantageous since it addresses the complexity and scalability issues of the 1SVM, especially when training with large-scale datasets. DBNs are appropriate feature detectors for anomaly detection, taking only raw (unlabelled) data to capture higher-order correlations among features, generating an accurate model, and imposing minimal computationally and memory complexity. Reducing the number of irrelevant and redundant features improves the scalability of a 1SVM for use with large training datasets containing high-dimensional records. In addition, by using a deep architecture, a DBN-1SVM can deliver better generalisation. Therefore, in the hybrid model more basic kernels, such as linear kernels, can be substituted for more complicated and computationally expensive ones, such as RBF kernels, with no loss in accuracy. Hence, our model offers an efficient, accurate and scalable anomaly detection approach that can be applied to large-scale and high-dimensional domains, such as the ones arise in the Internet of Things. Note that this empirical study has only been conducted on anomaly detection in sensor network datasets, and although similar results are potentially likely to arise in other domains due to the ability to use a linear kernel, no guarantee can be provided without further tests.

The experimental results on a wide range of benchmark datasets demonstrate that the AUC of DBN-1SVM outperforms standalone 1SVMs, where the improvement reaches around 20% in some datasets. Compared with an autoencoder, the difference is negligible, while the hybrid DBN-1SVM executes 3 times faster in training and 1000 times faster in testing.

A direction for future improvement is to adapt DBN-1SVMto on-line learning through incorporating incremental learning for 1SVMs. Currently the DBN is trained with mini-batches, which is suited to real-time training. However, 1SVMs are trained with a full-batch of records. Further, it will be interesting to apply DBN-1SVM to data streams, where the challenge would be how to maintain the accuracy of the model if normal behaviour evolves significantly over time.

Conflict of interest

None declared.

Acknowledgements

We thank the support from NICTA; the ARC Grants LP120100529 and LE120100129; University of Melbourne Early Career Research (ECR) grant; and the EU FP7 SocIoTal grant.

References

[1] ISSNIP, Internet of Things (IoT) for Creating Smart Cities. ⟨http://issnip.unimelb.edu.au/research_program/Internet_of_Things⟩, 2013.

[2] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of Things (IoT): a vision, architectural elements, and future directions, Future Gener. Comput. Syst. 29 (7) (2013) 1645–1660.

[3] R. Kassab, F. Alexandre, Incremental data-driven learning of a novelty detection model for one-class classification with application to high-dimensional noisy data, Mach. Learn. 74 (2) (2009) 191–234.

[4] S.M. Erfani, M. Baktashmotlagh, S. Rajasegarar, S. Karunasekera, C. Leckie, R1SVM: a randomised nonlinear approach to large-scale anomaly detection, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, pp. 432–438.

[5] Sarah M Erfani, Mahsa Baktashmotlagh, Sutharshan Rajasegarar, Vinh Nguyen, Christopher Leckie, James Bailey, and Kotagiri Ramamohanarao. R1STM: One-class Support Tensor Machine with Randomised Kernel. In Proceedings of SIAM International Conference on Data Mining (SDM), 2016.

[6] Sarah M. Erfani, Mahsa Baktashmotlagh, Masud Moshtahgi, Vinh Nguyen, Christopher Leckie, James Bailey, and Kotagiri Ramamohanarao. Robust Domain Generalisation by Enforcing Distribution Invariance. In Proceedings of International Joint Conference on Artificial Intelligence(IJCAI), 2016.

[7] A. Zimek, E. Schubert, H.-P. Kriegel, A survey on unsupervised outlier detection in high-dimensional numerical data, Stat. Anal. Data Min. 5 (5) (2012) 363–387.

[8] D.M. Tax, R.P. Duin, Support vector data description, Mach. Learn. 54 (2004) 45–66.

[9] D. Wang, D.S. Yeung, E.C. Tsang, Structured one-class classification, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 36 (6) (2006) 1283–1295.

[10] B. Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, 2001.

[11] S. Abe, Support Vector Machines for Pattern Classification, Springer, New York, 2005.

[12] L. Auria, R.A. Moro, Support Vector Machines (SVM) as a Technique for Solvency Analysis, Technical Report, Discussion Paper of DIW Berlin, German Institute for Economic Research, 2008.

[13] S. Vempati, A. Vedaldi, A. Zisserman, C. Jawahar, Generalized RBF feature maps for efficient detection, in: 21st British Machine Vision Conference, 2010, pp. 1–11.

[14] F.J. Huang, Y. LeCun, Large-scale learning with SVM and convolutional for generic object categorization, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 2006, pp. 284–291.

[15] D. Liu, H. Qian, G. Dai, Z. Zhang, An iterative SVM approach to feature selection and classification in high-dimensional datasets, Pattern Recognit. 46 (9) (2013) 2531–2537.

[16] Y. Bengio, Y. LeCun, Scaling learning algorithms towards AI, in: L. Bottou, et al., (Eds.), Large Scale Kernel Machines, 2007, pp. 1–41.

[17] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.

[18] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[19] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.

[20] Y. Liu, S. Zhou, Q. Chen, Discriminative deep belief networks for visual data classification, Pattern Recognit. 44 (10) (2011) 2287–2296.

[21] L. Cao, K. Chua, W. Chong, H. Lee, Q. Gu, A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine, Neurocomputing 55 (1) (2003) 321–336.

[22] J. Neumann, C. Schnörr, G. Steidl, Combined SVM-based feature selection and classification, Mach. Learn. 61 (1–3) (2005) 129–150.

[23] A. Widodo, B.-S. Yang, T. Han, Combination of independent component analysis and support vector machines for intelligent faults diagnosis of induction motors, Expert Syst. Appl. 32 (2) (2007) 299–312.

[24] A. Widodo, B.-S. Yang, Application of nonlinear feature extraction and support vector machines for fault diagnosis of induction motors, Expert Syst. Appl. 33 (1) (2007) 241–250.

[25] K.-Q. Shen, C.-J. Ong, X.-P. Li, E.P. Wilder-Smith, Feature selection via sensitivity analysis of SVM probabilistic outputs, Mach. Learn. 70 (1) (2008) 1–20.

[26] I. Guyon, Feature Extraction: Foundations and Applications, Springer, New York, 2006.

[27] T. Liu, A novel text classification approach based on deep belief network, in: 17th International Conference on Neural Information Processing: Theory and Algorithms (ICONIP), 2010, pp. 314–321.

[28] N. Bashah, I.B. Shanmugam, A.M. Ahmed, Hybrid intelligent intrusion detection system, in: World Academy of Science, Engineering and Technology, vol. 6, 2005, pp. 291–294.

[29] P. Hamel, D. Eck, Learning features from music audio with deep belief networks, in: International Society for Music Information Retrieval Conference (ISMIR), 2010, pp. 339–344.

[30] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.

[31] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, Neural Comput. 13 (7) (2001) 1443–1471.

[32] P. Laskov, C. Schäfer, I. Kotenko, K.-R. Müller, Intrusion detection in unlabeled data with quarter-sphere support vector machines, Prax. Inf. Kommun. 27 (4) (2004) 228–236.

[33] S. Rajasegarar, C. Leckie, J.C. Bezdek, M. Palaniswami, Centered hyperspherical and hyperellipsoidal one-class support vector machines for anomaly detection in sensor networks, IEEE Trans. Inf. Forensics Secur. 5 (3) (2010) 518–533.

[34] S. Rajasegarar, C. Leckie, M. Palaniswami, CESVM: Centered hyperellipsoidal support vector machine based anomaly detection, in: IEEE International Conference on Communications, 2008, pp. 1610–1614.

[35] S. Rajasegarar, C. Leckie, M. Palaniswami, J.C. Bezdek, Quarter sphere based distributed anomaly detection in wireless sensor networks, in: IEEE International Conference on Communications, 2007, pp. 3864–3869.

[36] C. Campbell, K. Bennett, A linear programming approach to novelty detection, in: Advances in Neural Information Processing Systems (NIPS), vol. 13, 2001, pp. 395–401.

[37] J.H. Ward, Hierarchical grouping to optimize an objective function, J. Am. Stat. Assoc. 58 (1–301) (1963) 236–244.

[38] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in Neural Information Processing Systems (NIPS), vol. 19, 2007, pp. 153–160.

[39] D. Wulsin, J. Gupta, R. Mani, J. Blanco, B. Litt, Modeling electroencephalography waveforms with semi-supervised deep belief nets: fast classification and anomaly measurement, J. Neural Eng. 8 (3) (2011) 036015.

[40] V. Nair, G.E. Hinton, 3D object recognition with deep belief nets, in: Advances in Neural Information Processing Systems (NIPS), vol. 21, 2009, pp. 1339–1347.

[41] Y. Tang, Deep learning using support vector machines, in: ICML Workshop on Challenges in Representation Learning, 2013.

[42] Y. Wang, D. Wang, Towards scaling up classification-based speech separation, IEEE Trans. Audio Speech Lang. Process. 21 (7) (2013) 1381–1390.

[43] G.E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput. 14 (8) (2002) 1771–1800.

[44] H. Larochelle, Y. Bengio, J. Louradour, P. Lamblin, Exploring strategies for training deep neural networks, J. Mach. Learn. Res. 10 (2009) 1–40.

[45] L.M. Manevitz, M. Yousef, One-class SVMs for document classification, J. Mach. Learn. Res. 2 (2002) 139–154.

[46] J.A. Lee, M. Verleysen, Nonlinear Dimensionality Reduction, Springer, New York, 2007.

[47] K. Brügge, A. Fischer, C. Igel, The flip-the-state transition operator for restricted Boltzmann machines, Mach. Learn. 93 (1) (2013) 53–69.

[48] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 27.

[49] L. Wang, U.T. Nguyen, J.C. Bezdek, C. Leckie, K. Ramamohanarao, iVAT and aVAT: enhanced visual analysis for cluster tendency assessment, in: Advances in Knowledge Discovery and Data Mining, 2010, pp. 16–27.

[50] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, D. Gunopulos, Online outlier detection in sensor data using non-parametric models, in: Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), 2006, pp. 187–198.

[51] G. Hinton, A practical guide to training restricted Boltzmann machines, in: Neural Networks: Tricks of the Trade, 2012, pp. 599–619.

[52] C.W. Hsu, C.-C. Chang, C.-J. Lin, et al., A practical guide to support vector classification. 〈https://www.cs.sfu.ca/people/Faculty/teaching/726/spring11/svmguide.pdf〉.

[53] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.

[54] S. García, F. Herrera, J. Shawe-taylor, An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons, J. Mach. Learn. Res. 9 (2008) 2677–2694.

[55] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. 180 (10) (2010) 2044–2064.

[56] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes, Pattern Recognit. 44 (8) (2011) 1761–1776.

[57] F. Wilcoxon, Individual comparisons by ranking methods, Biom. Bull. 1 (6) (1945) 80–83.

[58] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, 4th Edition, CRC Press, London, 2007.

[59] J.P. Shaffer, Modified sequentially rejective multiple test procedures, J. Am. Stat. Assoc. 81 (395) (1986) 826–831.

[60] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32 (200) (1937) 675–701.

[61] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.

Sarah M. Erfani is a Research Fellow in the Department of Computing and Information Systems at the University of Melbourne. Her research interests include large-scale data mining, machine learning, wireless sensor networks, and privacy-preserving data mining.

Sutharshan Rajasegarar is a Research Fellow in the Department of Computing and Information Systems at the University of Melbourne, and a Researcher in Machine learning at the National ICT Australia. His research interests include anomaly/outlier detection, machine learning, pattern recognition, data analytics, signal processing, wireless communication and Geographic Information Systems.

Shanika Karunasekera received the B.Sc degree in electronics and telecommunications engineering from the University of Moratuwa, Sri Lanka, in 1990, and the Ph.D. degree in electrical engineering from the University of Cambridge, UK, in 1995. From 1995 to 2002, she was a Software Engineer and a Distinguished Member of Technical Staff at Lucent Technologies, Bell Labs Innovations, USA. In January 2003, she joined the Department of Computer Science and Software Engineering, University of Melbourne as a Senior Lecturer. Her current research interests include decentralised algorithms for event and anomaly detection, and decentralised data aggregation.

Christopher Leckie is a Professor with the Department of Computing and Information Systems at the University of Melbourne. His research interests include using artificial intelligence for network management and intrusion detection, and data mining techniques such as clustering and anomaly detection.