Pattern Recognition 89 (2019) 161–171

Contents lists available at ScienceDirect

Pattern Recognition

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

Integration of deep feature extraction and ensemble learning for outlier detection

Debasrita Chakraborty[a], Vaasudev Narayanan[b], Ashish Ghosh[a,*]

[a] Machine Intelligence Unit, Indian Statistical Institute, 203 B. T. Road, Kolkata, 700108, India
[b] Department of Computer Science and Engineering, Indian Institute of Technology, Dhanbad, 826004, India

ARTICLE INFO

Article history:
Received 27 June 2018
Revised 7 December 2018
Accepted 2 January 2019
Available online 3 January 2019

Keywords:
Deep learning
Autoencoders
Probabilistic neural networks
Ensemble learning
Outlier detection

ABSTRACT

It is obvious to see that most of the datasets do not have exactly equal number of samples for each class. However, there are some tasks like detection of fraudulent transactions, for which class imbalance is overwhelming and one of the classes has very low (even less than 10% of the entire data) amount of samples. These tasks often fall under outlier detection. Moreover, there are some scenarios where there may be multiple subsets of the outlier class. In such cases, it should be treated as a multiple outlier type detection scenario. In this article, we have proposed a system that can efficiently handle all the aforementioned problems. We have used stacked autoencoders to extract features and then used an ensemble of probabilistic neural networks to do a majority voting and detect the outliers. Such a system is seen to have a better and reliable performance as compared to the other outlier detection systems in most of the datasets tested upon. It is seen that use of autoencoders clearly enhanced the outlier detection performance.

1. Introduction

Outliers play an important role in defining the nature of a dataset. They are certain interesting points in data that do not conform to the expected or the natural behavior of the dataset. They are usually anomalies, exceptions, discordant observations, surprises, peculiarities, aberrations, or contaminants in different application domains. An outlier is an observation in the data that is highly unlikely provided a model is built that generates the data [1]. In most of the practical cases, the model is abstract like that of finding a fraudulent credit card transaction in millions of genuine transactions. The data may also have multiple types of outliers like different types of intrusions in a network. One might consider the intrusions as a single outlier class and approach the problem in a binary or a multi-class fashion. However, the practicality of such an approach is questionable as intrusions are highly diverse and may have different reasons to appear in the data. There are many such cases which make single type outlier detection and multiple type outlier detection a crucial part of data analysis process. Figs. 1 and 2 show the two approaches. In the former one, the diversity of the outlier types are not considered and in the latter, the diversity of the outlier types are considered.

There is however a bottleneck that most of the algorithms get stuck at. Chances of finding an outlier in any dataset is extremely rare. In most of the cases, the number of samples from the outlier class is even below 10% of the total number of samples in the entire training set. Identifying them becomes one of the difficult problems in data analysis [2]. Sampling techniques are usually not preferred for outlier detection cases. Oversampling a minority (or an outlier) class or undersampling a majority (or the inlier) class usually affects the generalisation capability [3]. Moreover, the extreme imbalance (below 10%) is a major challenge for many algorithms. This is because the presence of an outlier is often misleading to algorithms like clustering, classification or regression. There may also be cases where the outliers may arise due to different reasons and may have diversity among them. When there are multiple types of outliers in the dataset, each having a different property, taking all the outlier classes as a single class may not make any sense. This differs from a multi-class imbalance problem as in this case the outliers consist of less than 10% of the entire dataset.

This article proposes and investigates a new supervised outlier detection framework inspired by the projection methodology through deep learning. It is shown analytically that the proposed method alleviates the aforesaid drawbacks of the existing standard approaches (we have not considered the unsupervised or semi-supervised outlier detection methodologies in order to give a fair comparison with the proposed method). We have done multiple experiments on several datasets to prove whether the non-linear

* Corresponding author.
E-mail address: ash@isical.ac.in (A. Ghosh).

https://doi.org/10.1016/j.patcog.2019.01.002
0031-3203/

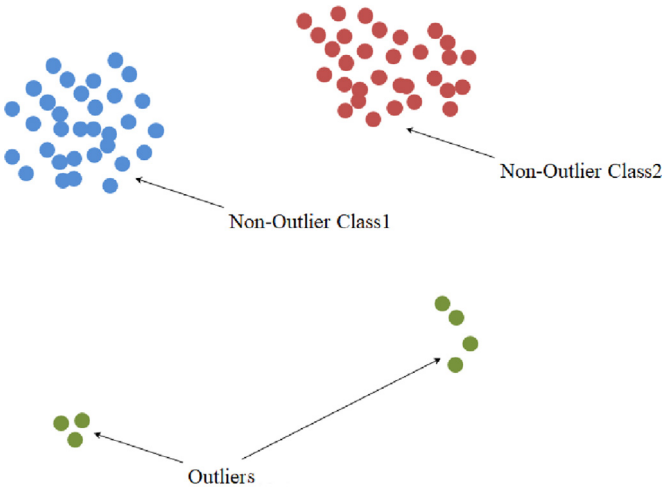

Fig. 1. Multi-class single outlier scenario.

transformation given by the auto-encoders or the probabilistic networks used make the method better. It is seen that the proposed system gives a good performance on almost all the datasets experimented upon. The proposed method is also capable of handling the diversity in the multiple types of outliers in a dataset.

The contributions are as follows:

- A novel outlier detection framework using the projection principles of stacked autoencoders and probabilistic neural networks.
- An efficient technique to identify outliers in both single type outlier as well as multiple outlier type datasets.
- This article also highlights how the use of deep stacked autoencoders can enhance the performance of the standard outlier detection techniques.

The paper is organised as follows. Section 2 surveys some of the existing and popular methods for outlier detection. Section 3 describes the proposed framework. The empirical investigation of the proposed method is described in Section 4. Section 5 gives a statistical comparison of the methods discussed, while Section 6 draws the conclusions.

2. Related works

The existing outlier detection methods [4] make different assumptions and hence differ in the way they detect the outliers. Researchers have proposed an abundance of outlier detection techniques [5,6]. Standard outlier detection techniques mostly include distance-based methods [7,8], and density-based methods [9]. The problem of outlier detection is usually handled in three types of learning scenarios:

(i) Unsupervised: Learning only from the inlier class (no information of the outlier class is provided),
(ii) Supervised: Learning from the inlier as well as outlier classes, and
(iii) Semi-supervised: Learning only from the inlier class and some unlabeled data (which may or may not be outliers).

Unsupervised methods do not use any information about the outlier class. They include one class classifiers [10,11], density-based methods [7,8], clustering based methods [12] etc. One class classifiers try to build a boundary around the inlier class and any sample that lies outside the boundary is classified as an outlier. Density based methods find the density at each region in the feature space and when the density is found to be low, the region is decided to belong to the outliers. Clustering-based approaches [12] identify an object as an outlier if it does not belong to any cluster or if the size of the cluster is too small. All the unsupervised methods are advantageous in the sense that they do not need any outlier information and are hence robust. However, it is to be noted that most of these methods usually set at the outliers to a single group. Supervised methods [6] have an edge on unsupervised methods since they use the outlier information to refine their boundaries. However, unsupervised methods may catch any new anomaly that the supervised method may not as it has not been trained upon that particular type of outlier. Semi-supervised methods [13] are more elegant than the former two where they make use of unlabelled data and train only on inlier data. There is however a claim that using a very small fraction of outliers in the training data is always good for the flexibility of boundaries [11]. Hence, in this article, we have considered only the supervised outlier detection methodologies for a fair comparison with the proposed method.

The most popular of the supervised outlier detection algorithms is the classic and simple k-nearest neighbor (kNN) [14]. The popular LOF (local outlier factor) method [6,15] also makes use of kNN search for each point. It assigns an LOF score to each sample in the dataset irrespective of the data skewness. However, most of the state of the art nearest neighbour based outlier detection methods [16,17] are sensitive to the parameter k. It means that small change in k can lead to higher misclassification rates [18]. So, we have considered the basic kNN only for comparison purposes as the state of the art nearest neighbour based outlier detection methods are basically based on kNN itself.

Support vector machines (SVMs) have also been extended to one-class learning [19]. Support Vector Data Description (SVDD) [11] has been introduced as a generative approach to the standard SVM. The idea is to build a hypersphere that encompasses all the inlier data and the samples that lie outside are considered outliers. It was shown [11], that SVDD performance are almost comparable to Gaussian, Parzen density estimators and Nearest Neighbor methods. However, it is noteworthy that methods like SVDD (which are strictly one class classifiers) assume an inevitable presence of a fraction of outlier samples in the legitimate non-outlier data [11] to make the decision boundary flexible. However, SVMs are not suited for datasets with large number of samples and take a long time to train. Taking linear kernels may reduce the time complexity, but it makes no sense to use a linear kernel for datasets that have high complexity.

The *i*Forests (Isolation Forests) method [20] has also gained reputation in the outlier detection community. This strategy is exceptionally helpful and is unique compared to every single existing technique. It isolates the outliers considering them to be far from the rest of the observations. It builds up a Random Forest model with a few decision trees using only a small fixed sized subsamples, irrespective of the size of the dataset. Therefore, when a forest of such small random decision trees collectively give shorter path lengths for any particular sample, it tends to be an outlier. However, the method's inability to detect outliers in datasets with a high percentage of irrelevant features causes it to fail in most of the cases. It was experimentally found in [21] that dimensionality reduction enhances the classification performance of such methods to a much significant level.

Deep neural networks (DNNs) are being extensively researched to give an answer to the problem of outlier detection [13]. Since, a normal multi-layer perceptron is affected by the masking and swamping effects of outliers, it is difficult for any feedfoward multi-layer network to handle outliers. Researchers have been using self organising maps to handle such problems in unsupervised fashion [22]. Recently, outlier detection have been achieved using variational auto-encoders [23]. However, these methods can only

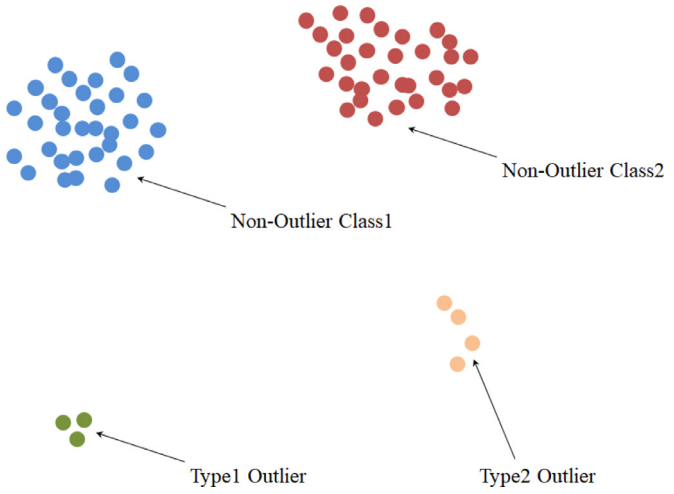

Fig. 2. Multi-class multiple outlier scenario.

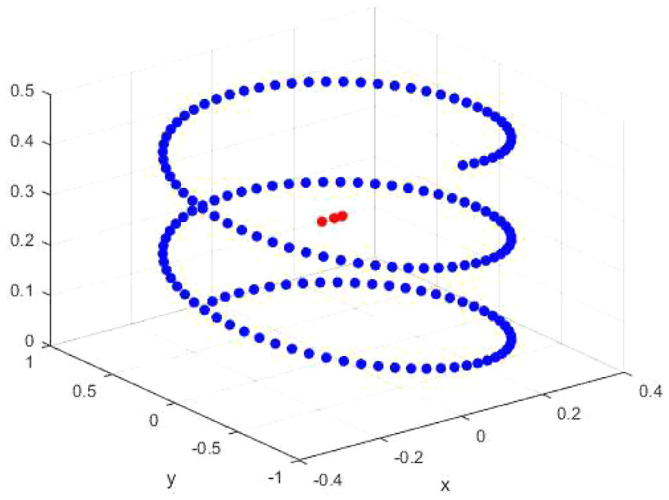

Fig. 3. An example where the repetitive structures in the underlying manifold of the non-outlier class can wash out outlier effects.

be used for datasets with a substantially larger number of samples. Also, DNNs being just an extension of the classical neural networks also suffer from similar masking and swamping effects of outliers. In contrast, networks like radial basis functions (RBF) neural networks or probabilistic neural networks (PNN) are structurally more suitable for outlier detection [24].

There are many methods which can detect outliers [25] in a multi-class dataset. These methods assume that all the outliers should always belong to a single class, though practical data may have different types of outliers arising out of different causes. Such methods can therefore be regarded as "multi-class single outlier type detection techniques". We have introduced the problem in a different way where the outlier class too can have multiple subtypes along with the multiple types of non-outlier classes. Outlier classes and under-represented classes need to be clearly differentiated in that case. Those classes which have rare occurrences in the data (10% or less) and do not comply with the normal observations are to be treated as outliers; whereas the classes which are just under-represented (more than 10% but lesser in comparison to the majority class) should be treated as a normal class in a multi-class dataset. Also, we need to consider the fact that there may be more than one outlier classes originating out of different abnormal causes. The classical Random Forests [26] is capable of handling such a situation. It is due to the decision tree technique, it is based upon, which isolates the anomalous observations into small leaves.

For many complex problems, a linear transformation fails to capture the underlying manifold for each class in the data. So, we resort to non-linear dimensionality reduction techniques. Researchers in [27] have shown that it is a property of auto-encoders to map repetitive structures to a similar range. In this way, autoencoders can efficiently capture the class-specific data manifold while eliminating the redundancy. For problems like anomaly detection, the redundancies are inevitable as the non-outlier data forms the majority in the data and may form repetitive structures in the underlying manifold which can often make it difficult to differentiate the outliers. For reference, Fig. 3 can give some idea as to how the repetitive structures in the underlying manifold can wash out outlier effects. The red points are inside the region of the helical structure. But, the distribution followed by the non-outlier class is different from the outliers. However, classifiers like *k*NN may consider the outlier points nearer to an outlier point than the non-outlier point on the opposite side of the helix. In such cases, autoencoders may find the repetitive structures in the non-outlier class and can map it in almost the same range and outliers to a different range. Autoencoders can enhance the outlier detection mechanism in such cases.

The proposed method uses a non-linear transformation by stacked autoencoders [28] and then uses an ensemble of probabilistic neural network to model the outlier class. In doing so, it can efficiently identify outliers in both single type outlier dataset as well as multiple type outlier dataset with better performance than the standard methods used. The proposed method, hence, falls into a category of multi-class multiple outlier type detection technique. There is a lack of dedicated methods in the literature that can handle the outlier detection problem in both multi-class single outlier type detection and multi-class multiple outlier type detection.

3. Proposed method

We have proposed a novel outlier detection technique using deep stacked autoencoders and probabilistic neural networks. Such a system is shown to provide better results than the state of the art techniques. The proposed method efficiently handles both the single type outlier cases as well as multiple type outlier cases. This section discusses the concepts and strategies employed in the proposed method. In Sections 3.1 and 3.2 we briefly explain the working of a probabilistic neural network and the autoencoder networks. In Section 3.3 we introduce the proposed method and the various strategies involved to perform efficient outlier detection.

3.1. Probabilistic neural networks

A Probabilistic Neural Network (PNN) is a representation of a statistical algorithm called Kernel Discriminant Analysis (KDA). The operations are organized into a multilayered feedforward network with mainly three layers namely the input, pattern and summation layers [Fig. 4]. (In some cases, there is also an output layer, which we have omitted in our model, as we are interested in the immediate class scores rather than hard class labels.) In a way it estimates the probability density function (*pdf*) of classes by using the samples of the training set. In the pattern layer, each node calculates the *pdf* for *each* training sample according to Eq. (1).

$$g(\mathbf{x}) = \frac{1}{\sigma}\mathcal{F}\left(\frac{\mathbf{x}-\mathbf{x}_i}{\sigma}\right) \tag{1}$$

where $\mathbf{x}$ is the unknown (input), $\mathbf{x}_i = i^{th}$ sample, $\mathcal{F}()$ is the weighting function and σ is the smoothing parameter. In the pattern layer, the nodes (corresponding to the training pattern) are

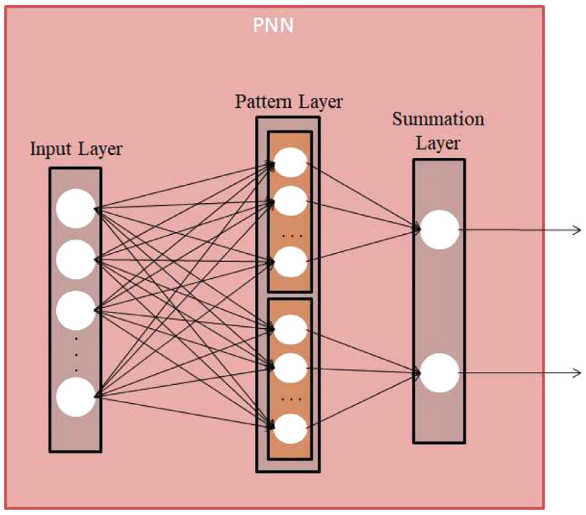

Fig. 4. A schematic diagram of probabilistic neural network.

grouped according to the class of the training sample. A single group accounts operation for the next summation node.

If there are C classes, the k^{th} summation node sums the values received from the pattern nodes in the k^{th} class, called mixed Gaussians or Parzen windows. The sums are defined by

$$f_k(\mathbf{x}) = \frac{1}{n\sigma}\sum_{i=1}^{n_k}\mathcal{F}\left(\frac{\mathbf{x}-\mathbf{x}_i}{\sigma}\right), \tag{2}$$

where n_k is the number of samples in the k^{th} class.

The output from the k^{th} node in the summation layer corresponds to the k^{th} class. As newer training patterns arrive, the estimate of $f_k(\mathbf{x})$ is refined without the need of any retraining. As the number of samples increase, each summation node in the PNN tends to get closer to the actual underlying class density functions.

Any input vector $\mathbf{x}$ is put through all the functions $f_k(\mathbf{x})$ and the maximum a posteriori probability (MAP) value of all the $f_k(\mathbf{x})$ decides the class.

Usually, the weighting function $\mathcal{F}()$ is chosen to be a radial basis function (RBF) (also called a kernel function). The radial basis function is so named because the radial distance is the argument to the function. The farther a training sample is from the new unknown point, the less influence it has. A PNN is insensitive to outliers and is hence a very good candidate for outlier detection. PNNs also have an added benefit as new patterns arrive, they are stored into the net. The network improves its generalization and the decision boundary can get more complex. If the number of training patterns become large, the value of σ can be reduced without retraining the network. It was shown by researchers [11] that in case of high sample size, Parzen density estimators are to be preferred over SVMs. Since, each subnetwork of a PNN act as the Parzen density estimator for the particular class, it can easily be used for outlier detection purposes.

However, it was observed that use of PNN alone could not provide better results. Moreover, similar to kNN, such a system has a large storage requirement. We have tried to tackle such a high storage requirement by dimensionality reduction using stacked autoencoders (SAE).

3.2. Autoencoders

An autoencoder (AE) is a type of multi-layer neural network that is trained to reconstruct the input as closely as possible in its output nodes [Fig. 5].

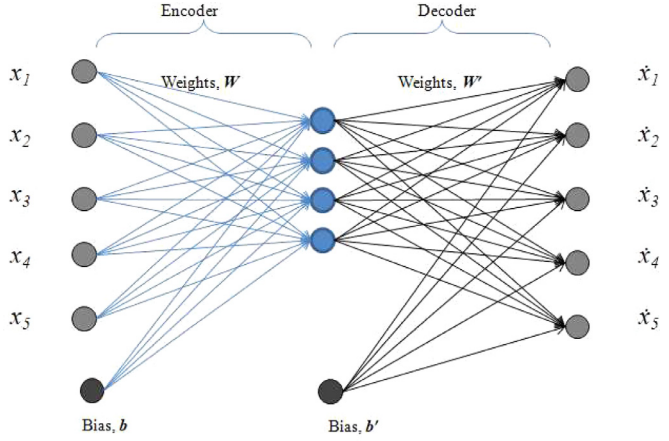

Fig. 5. A schematic diagram of an under-complete autoencoder.

While stacking the autoencoders we use the encoder part and the hidden nodes become the input to the next autoencoder network. If the nodes in the hidden layers are lesser in number, they represent a reduced representation of the input and are hence called under-complete autoencoders. Such a concept of reduced representation, makes it a natural approach in outliers detection. The basic motivation is that usually the outliers are much harder to be accurately represented in such reduced representation and hence stand out of the normal data distribution.

3.3. Stacked autoencoder probabilistic neural network (SAE-PNN)

Most of the outlier detection methods, employ semi-supervised [5] or unsupervised learning [6,29]. The semi-supervised models are usually trained using only the inlier class because abnormal or outlier points occur rarely and hence are not easily available. Outliers being only a very small fraction of the data, can never be sufficient enough to construct a binary classifier. However, they can be used to efficiently refine the decision boundary that separates the inlier samples. In the proposed method, we have trained the stacked autoencoders using both the labelled as well as unlabelled samples. However, for the final training of the ensemble of PNNs, we have used a tiny fraction of the labelled outlier samples along with the majority of the inlier samples. This inclusion of outliers in the training set was done to incorporate a refinement of the model. In case of outlier detection, we need to make a trade-off between generalization (keeping the inliers inside the decision boundary) and specialization (excluding the outlier patterns outside the boundary). Most of the models used for outlier detection cannot specialize from data unless they are given an internal bias. The proposed model alleviates these pitfalls. However, there were other factors too which needed to be considered.

The peak of the radial basis function in a PNN is always centered at the point it is weighting. The smoothing parameter (σ) determines the spread of the RBF. The primary work of training a PNN is selecting the optimal sigma values to control the spread. Extremely large values of σ cause the model to under-fit and extremely small values cause the model to over-fit. However, it was seen that for imbalanced datasets, lower values of sigma gave lower false alarms and higher values of sigma produced lower misses. Since, both the factors are important in case of efficient anomaly detection, it motivated us to use an ensemble of PNNs, each having a different sigma value, to get the best of the both.

Suppose the outlier set is $\mathcal{O}$ and the inlier set is $\mathcal{I}$ and the original training dataset is $\mathcal{S}$. Then,

$$\mathcal{O} \cup \mathcal{I} = \mathcal{S} \tag{3}$$

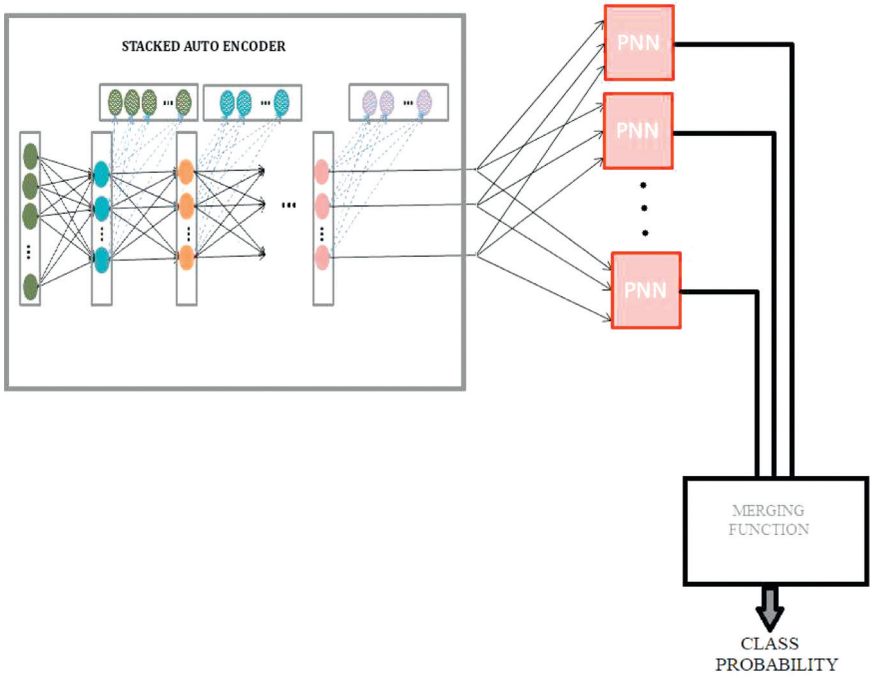

Fig. 6. A diagrammatic representation of the proposed framework. (The dashed lines suggest that the decoder part of the autoencoder was not used after feature extraction at each stage.)

The set $\mathcal{I}$ is randomly divided into N subsets such that

$$\mathcal{I}_1 \cup \mathcal{I}_2 \ldots \mathcal{I}_N = \mathcal{I} \tag{4}$$

Here, N is the number of PNNs we have used in the ensemble. Each of the i^{th} PNNs is trained with entire set $\mathcal{O}$ and the set $\mathcal{I}_i$. For multi-class multiple outlier type dataset, the majority class training samples are divided into N subsets and the minority classes as well as the outlier classes are taken as a whole.

However, such methods perform better when any dimensionality reduction has been performed on the datasets [21]. We know that autoencoders provide a non-linear transformation and perform dimensionality reduction efficiently [30]. It was empirically found that stacking more (usually greater than two) number of autoencoders in the feature extraction stage resulted in better outlier detection rates. In Section 4.3, we have shown this. A diagrammatic representation of the proposed framework has been shown in Fig. 6.

3.3.1. Stacked autoencoder

Let us consider after feature extraction using the stacked autoencoder (SAE), each feature vector gets a transformation $SAE(\mathbf{x})$, where,

$$SAE(\mathbf{x}) = \mathcal{E}_1(\mathcal{E}_2(\ldots \mathcal{E}_h(\mathbf{x}))). \tag{5}$$

Here $\mathcal{E}_i()$ represents the encoder function at the i^{th} stacked autoencoder and h represent the number of hidden layers used in SAE.

3.3.2. Probabilistic neural network

The intermediate RBF scores of $\mathbf{x}$ belonging to each of the k classes in the n^{th} PNN was given according to

$$l_k^n((x)) = f_k^n(SAE(x)). \tag{6}$$

3.3.3. Merging function

If there were N PNNs used in the ensemble, the final RBF score $\mathbf{x}$ belonging to each of the k classes in the entire ensemble could be obtained according to

$$l_k(\mathbf{x}) = \sum_{n=1}^{N} (l_k^n((x))). \tag{7}$$

Now, the classification scores of $\mathbf{x}$ belonging to each of the k classes is calculated as

$$s_k(\mathbf{x}) = \frac{l_k(\mathbf{x}) - min(l_k(\mathbf{x}))}{max(l_k(\mathbf{x})) - min(l_k(\mathbf{x}))}. \tag{8}$$

3.3.4. Class probability

The probability of $\mathbf{x}$ belonging to the k^{th} class is obtained as,

$$p_k(\mathbf{x}) = softmax(s_k(\mathbf{x})). \tag{9}$$

The node that gives the maximum probability is the true class label for the unknown sample $\mathbf{x}$.

4. Experiments

This section is committed to illustrate the abilities of the proposed method in various situations of outlier detection. We have done some thorough experiments as to how the results vary with values of σ and the depth of the SAE network used. We have given a comparative analysis of the performance of our proposed method with some of the standard methods used in outlier detection both with SAE and without SAE to clearly study the effects of using SAE in outlier detection performance. Finally, we also show that the proposed method is capable of handling multiple outlier type dataset.

For fair comparison, the value of k in k-NN was chosen using cross-validation. The number of predictors for both *i*Forests and Random Forests were taken at a default value of 100. The SAE was trained using 'adadelta' optimiser and mean squared error was used. The model was pre-trained for 200 epochs.

4.1. Datasets used

The datasets used in the experimental analysis of the proposed method are all highly skewed in nature. Table 1 shows the various

Table 1
List of single outlier type data sets used for experimental purpose.

Dataset	Number of features	Number of samples	Percentage of outliers
PenDigits [31]	16	6870	2.27%
OptDigits [32]	64	5216	3%
Forest Cover [33]	10	286,048	0.9%
MNIST [34]	100	7603	9.2%
HTRU2 [35]	8	17,898	9.2%
SatImage-2 [32]	36	5803	1.2%
Credit card [36]	29	284,807	0.172%
Speech [37]	400	3686	1.65%

Table 2
Number of samples in each class for multiple outlier type datasets.

	Datasets			
	Statlog (shuttle)	Page blocks	ANN thyroid disease	Wine quality (white)
Class 1	45586 (78.6%)	4913 (89.8%)	93 (2.47%)	20 (0.41%)
Class 2	50 (0.09%)	329 (6%)	191 (5.06%)	163 (3.32%)
Class 3	171 (0.29%)	28 (0.5%)	3488 (92.47%)	1457 (29.74%)
Class 4	8903 (15.35%)	88 (1.6%)		2198 (44.9%)
Class 5	3267 (5.63%)	115 (2.1%)		880 (17.96%)
Class 6	10 (0.017%)			175 (3.57%)
Class 7	13 (0.023%)			5 (0.1%)

Table 3
Confusion matrix for the non-outlier class.

Actual class	Predicted class	
	Inliers	Outliers
Inliers	Hit (True positive)	False alarms (false negative)
Outliers	Missed alarms (False positive)	Outlier hit (True negative)

single outlier type datasets used and their properties. The main complexity of each of the datasets lies in the fact that the outlier class is less than 10% of the total samples. We have used 10 fold cross validation to analyse the performance on each of these datasets. To analyse the performance of the proposed method on multiple outlier type datasets, we have used four highly skewed datasets namely Statlog (Shuttle), Page Blocks, ANN-Thyroid Disease and Wine Quality (White) from the UCI Machine Learning repository. For the Statlog (Shuttle) and ANN-Thyroid Disease datasets, the training and the testing sets were already separated. For the other two datasets, we have used 70% for training and 30% for testing, so that the outlier fraction remains the same in both the training and the test set. As can be seen from Table 2, many classes constitute less than 10% of the data and are therefore rarely occurring. However, each of these classes have a different reason to appear in the data. For example, in the Page Blocks dataset there are five classes which represent text, horizontal line, graphic, vertical line and picture, respectively. Majority of the samples belong to text, but the other four classes vary in properties and are not similar to each other to be put into a single outlier class. In this case, there are multiple classes which do not conform to the majority (which in this case is 'text') of the data.

4.2. Dependence on σ

For any outlier detection problem, the confusion matrix for the non-outlier class looks like Table 3. FNR (false negative rate) suggests the rate as to how often the classifier predicts a sample as an outlier when actually it is not. It is given by

$$FNR = \frac{False\ Negatives}{True\ Positives + False\ Negatives}. \tag{10}$$

TNR (true negative rate) suggests the rate as to how often the classifier predicts a sample as an outlier when actually it is an outlier. It can be represented as

$$TNR = \frac{True\ Negatives}{False\ Positives + True\ Negatives}. \tag{11}$$

Any good outlier detection method should be able to give high TNR and low FNR. So, we can say, the detection rate (DR), where

$$DR = (1 - FNR) \times TNR \tag{12}$$

is high. It is seen (Fig. 7) that the value of σ has an optimal region (shaded by green) with respect to the DR curve. Within that region of σ, it is seen that for imbalanced datasets, lower values gave lower false alarm rates (FNR) and higher values gave a lower outlier miss rates (FPR) so that the detection rate is quite high. Fig. 8 shows how the ensemble of PNNs each having a different sigma value helps us get the best of both. In order to choose the sigma value for each PNN, we have first noted down the plateau region of the sigma for a single PNN as shown in Fig. 7. Now, if there are N PNNs, the plateau region of sigma is divided equally into N equispaced values. Each of these values are then taken as sigma for each of the N PNNs.

Table 4
Autoeencoder configuration used for each dataset.

Dataset	Number of hidden layers	Nodes in each hidden layer
PenDigits	9	5
OptDigits	6	10
Forest Cover	9	4
MNIST	4	20
HTRU2	16	7
SatImage-2	20	10
Credit card	12	5
Speech	3	100

4.3. Dependence on number of autoencoders stacked

In this section we have shown an empirical investigation of the effect of stacking more (usually greater than two) number of autoencoders in the feature extraction stage. Table 4 gives the network specifications of stacked autoencoders used for each dataset. To provide a comparison regarding the use of autoencoders with multiple PNNs against using only multiple or single PNNs in the outlier detection problem, we have given a curve that depicts the variation of the ROC-AUC (Area Under the Curve of Receiver Operating Characteristics) with respect to the number of hidden layers. Fig. 9 shows that the use of more hidden layers in the autoencoders have significantly enhanced the performance. However, it is also seen that there is an optimal depth beyond which the performance has decreased too. This suggests that use of deep neural networks certainly enhances the performance of the model, but going too deep is not always the best option.

4.4. Performance of the proposed method on single outlier type datasets

This experiment reveals the comparison of outlier detection performance of the proposed method with some of the popular methods in literature. We have shown how the use of stacked autoencoders enhances the performance of those methods too.

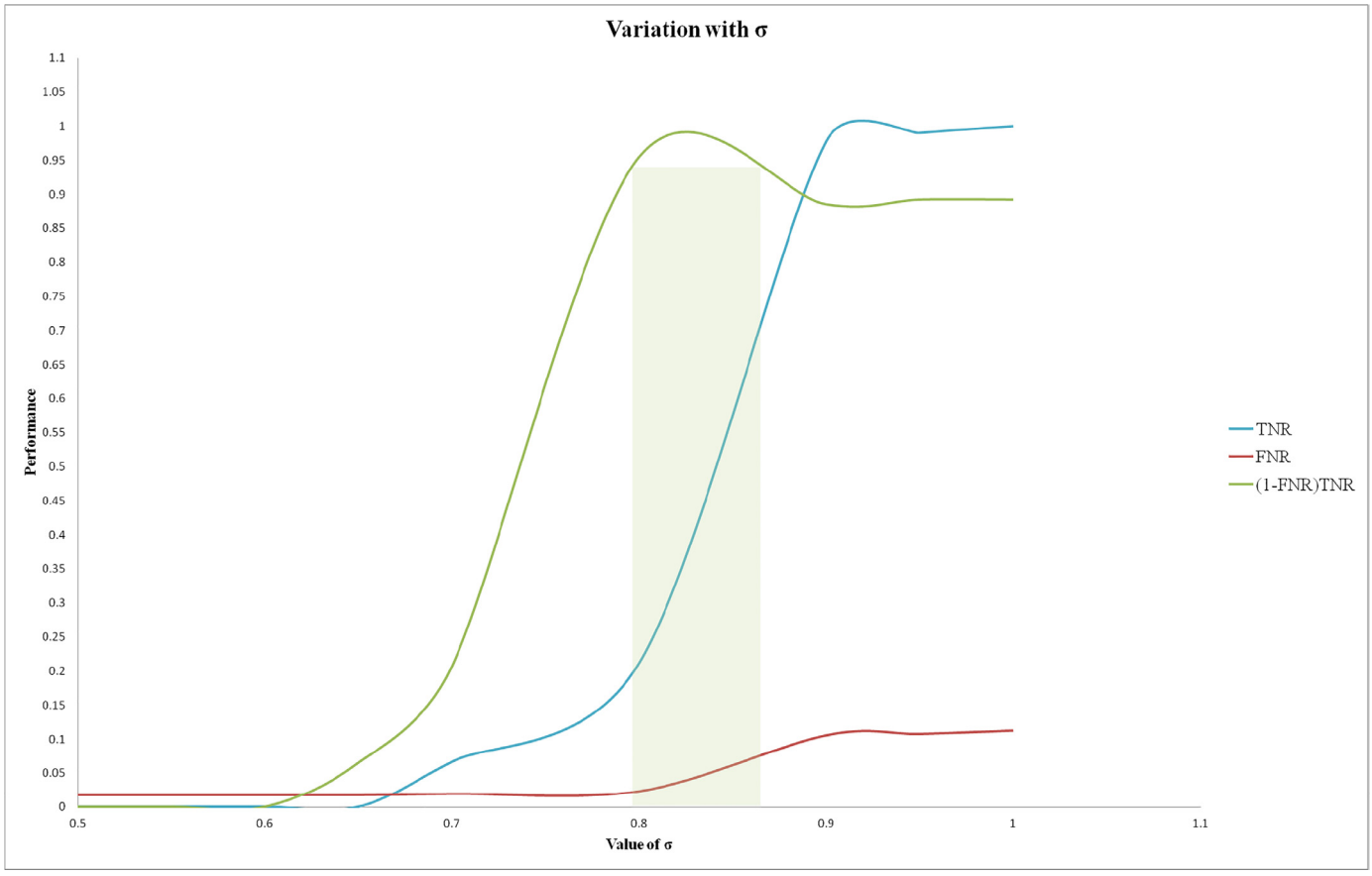

Fig. 7. Variation of various performance measures (*TNR*, *FNR* and $(1 - FNR) \times TNR$) with respect to σ. (Calculations shown are for OptDigits [32] dataset with the autoencoder configuration 64-10-10-10-10-10-10).

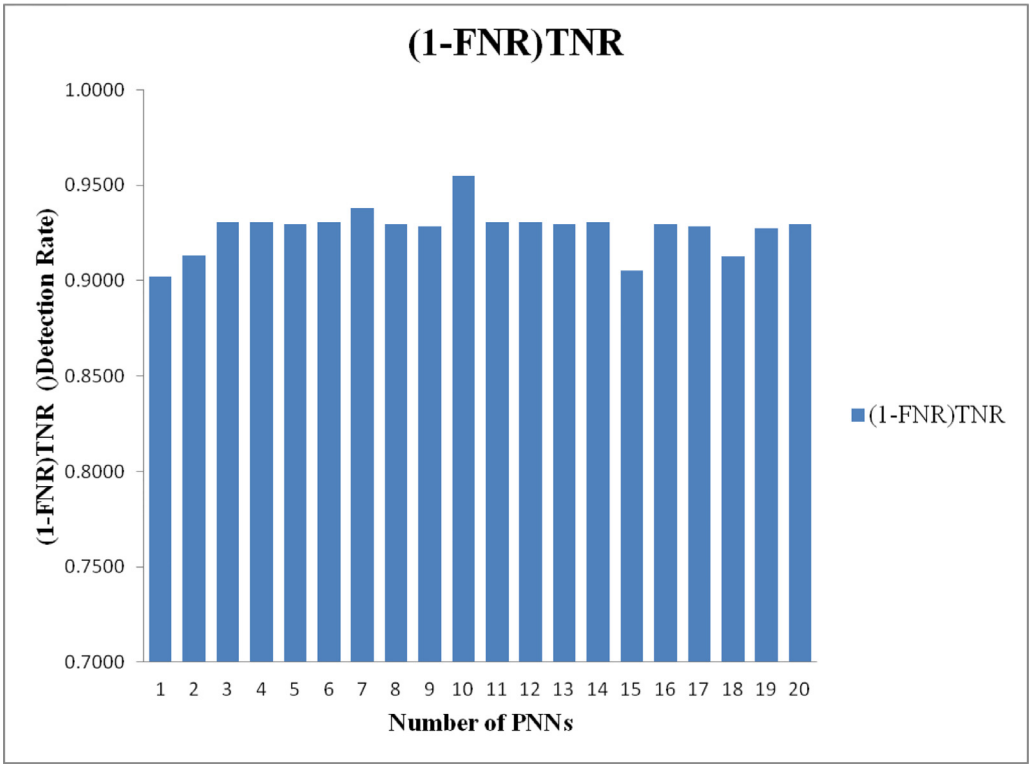

Fig. 8. Variation of the detection rate ($(1 - FNR) \times TNR$) with respect to number of PNNs used. (Calculations shown are for OptDigits [32] dataset with the autoencoder configuration 64-10-10-10-10-10-10).

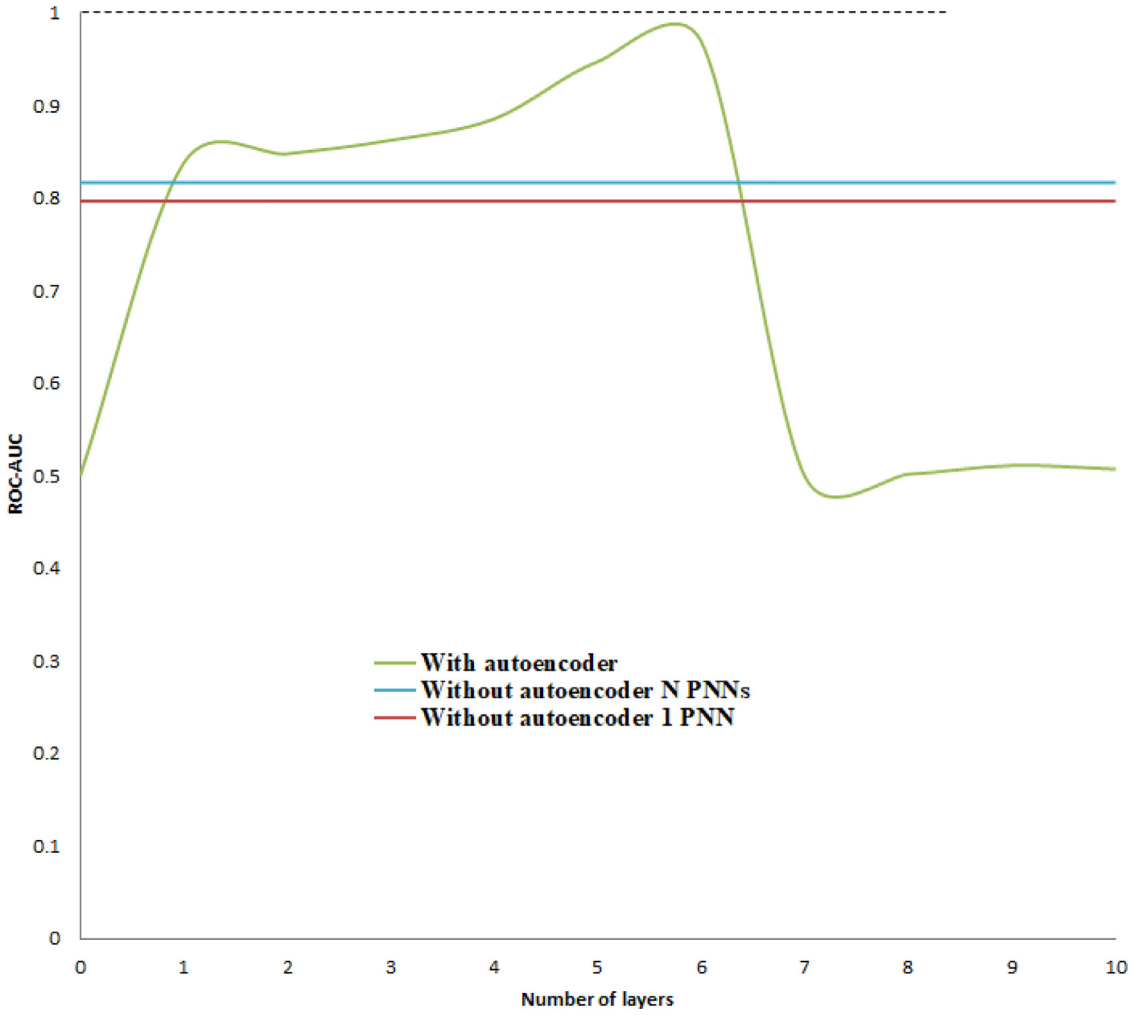

Fig. 9. Variation of the ROC-AUC with respect to the number of hidden layers used. (Calculations shown are for OptDigits [32] dataset where the number of hidden neurons in each layer is 10).

Table 5
Performance of different methods on single outlier type data sets.

Dataset	Performance metrics	SAE-PNN (proposed)	SAE-*k*NN	SAE-*i*Forests	PNN	*k*NN	*i*Forests
PenDigits	ROC-AUC	**0.9923**	0.9619	0.9370	0.8880	0.9848	0.8699
	F-Score	**0.9656**	0.9563	0.7339	0.6917	0.9387	0.7268
	G-Mean	**0.9659**	0.9563	0.7517	0.7093	0.9409	0.7368
OptDigits	ROC-AUC	**0.9684**	0.9000	0.9056	0.8905	0.8467	0.6072
	F-Score	0.8951	0.9008	0.8925	0.8352	**0.9077**	0.6423
	G-Mean	0.8976	0.9008	0.8925	0.8368	**0.9101**	0.6433
Forest Cover	ROC-AUC	**0.9324**	0.8118	0.9205	0.9054	0.8453	0.7823
	F-Score	**0.9335**	0.8931	0.7377	0.9216	0.9131	0.6393
	G-Mean	**0.9335**	0.8977	0.7527	0.9244	0.9160	0.6503
MNIST	ROC-AUC	**0.9454**	0.9381	0.9414	0.8292	0.9366	0.6786
	F-Score	**0.9453**	0.9362	0.9410	0.8254	0.9350	0.6782
	G-Mean	**0.9454**	0.9372	0.9412	0.8273	0.9358	0.6784
HTRU2	ROC-AUC	**0.9262**	0.8059	0.9103	0.9216	0.8233	0.7754
	F-Score	**0.9209**	0.8723	0.9040	0.9103	0.8849	0.8015
	G-Mean	**0.9210**	0.8753	0.9040	0.9104	0.8873	0.8019
SatImage-2	ROC-AUC	**0.9589**	0.8833	0.9537	0.8661	0.9167	0.9564
	F-Score	**0.9432**	0.9372	0.7444	0.9088	0.9279	0.7247
	G-Mean	**0.9433**	0.9389	0.7630	0.9099	0.9303	0.7469
Credit card	ROC-AUC	**0.9421**	0.8821	0.9153	0.8000	0.8475	0.9008
	F-Score	0.7760	**0.9249**	0.7269	0.7896	0.9078	0.6510
	G-Mean	0.7884	**0.9260**	0.7428	0.7975	0.9101	0.6776
Speech	ROC-AUC	**0.8769**	0.6797	0.7448	0.7738	0.5400	0.4931
	F-Score	**0.8007**	0.7905	0.6644	0.6329	0.6975	0.4959
	G-Mean	**0.8038**	0.8012	0.6683	0.6437	0.7292	0.4959

Table 6
Confusion matrices.

(a) Confusion matrices for SAE-PNN

Actual class	Predicted class				
	Class 1	Class 2	Class 3	Class 4	Class 5
Class 1	1302	24	40	1	36
Class 2	47	90	5	1	3
Class 3	1	0	6	0	16
Class 4	7	1	12	34	1
Class 5	0	0	0	0	16

(b) Confusion matrix for SAE-kNN ($k = 35$)

Actual class	Predicted class				
	Class 1	Class 2	Class 3	Class 4	Class 5
Class 1	1389	5	0	1	8
Class 2	73	73	0	0	0
Class 3	5	0	0	1	17
Class 4	43	1	0	11	0
Class 5	4	0	0	0	12

(c) Confusion matrix for SAE-Random Forests

Actual class	Predicted class				
	Class 1	Class 2	Class 3	Class 4	Class 5
Class 1	1388	2	0	1	12
Class 2	77	67	0	1	1
Class 3	6	0	1	2	14
Class 4	35	0	0	20	0
Class 5	5	0	0	0	11

(d) Confusion matrix for PNN

Actual class	Predicted class				
	Class 1	Class 2	Class 3	Class 4	Class 5
Class 1	1043	4	277	5	74
Class 2	48	62	12	19	5
Class 3	0	0	10	0	13
Class 4	0	0	8	45	2
Class 5	0	0	0	0	16

(e) Confusion matrix for kNN ($k = 35$)

Actual class	Predicted class				
	Class 1	Class 2	Class 3	Class 4	Class 5
Class 1	1396	4	0	0	3
Class 2	81	65	0	0	0
Class 3	20	0	0	0	3
Class 4	52	0	0	0	3
Class 5	11	0	0	0	5

(f) Confusion matrix for Random Forests

Actual class	Predicted class				
	Class 1	Class 2	Class 3	Class 4	Class 5
Class 1	1089	37	0	255	22
Class 2	40	102	0	2	2
Class 3	0	0	0	18	5
Class 4	4	1	0	49	1
Class 5	3	0	0	8	5

Table 7
Performance of different methods on multiple outlier type data set.

Dataset	Performance metrics	SAE-PNN (proposed)	SAE-kNN	SAE-Random Forests	PNN	kNN	Random Forests
Statlog (shuttle)	ROC-AUC	**0.9713**	0.8988	0.9176	0.9326	0.8988	0.8876
	F-Score	**0.9342**	0.9224	0.9317	0.8459	0.9224	0.8968
	G-Mean	**0.9349**	0.9228	0.9318	0.8496	0.9228	0.8969
Page blocks	ROC-AUC	**0.9554**	0.8276	0.8139	0.8328	0.8239	0.8501
	F-Score	**0.6400**	0.5392	0.6226	0.6068	0.3897	0.4550
	G-Mean	**0.6414**	0.5422	0.6388	0.6107	0.3922	0.4603
ANN-thyroid disease	ROC-AUC	**0.9951**	0.9929	0.9921	0.8715	0.8630	0.9923
	F-Score	0.8645	0.9041	0.9075	0.6502	0.6563	**0.9464**
	G-Mean	0.8685	0.9041	0.9076	0.6548	0.6747	**0.9465**
Wine quality white	ROC-AUC	**0.7900**	0.7172	0.7072	0.7867	0.7131	0.7050
	F-Score	0.3527	0.2246	0.3884	**0.4143**	0.2272	0.3965
	G-Mean	0.3544	0.2246	0.3917	**0.4165**	0.2272	0.3990

Table 5 shows a comparative analysis for the datasets mentioned in Table 1.

In all the datasets, the proposed method gives the highest ROC-AUC [1]. In majority of the datasets, the F-Score [38] and G-Mean [38] of the proposed method is competitive with the best performance.

4.5. Performance of the proposed method on multiple outlier type dataset

To validate the efficiency of the proposed method on multiple outlier type dataset, we have used four highly skewed datasets.

To show the efficacy of the proposed method, we have also provided the comparison of confusion matrices obtained using each method for the Page Blocks dataset. It is evident from Table 6 that most of the classifiers failed to capture Class 3 (which belongs to only 0.5% of the data). Since, for kNN, $k = 35$ gave the highest ROC value, we have reported the results taking $k = 35$ only for the comparison of confusion matrices. A quantitative comparison have also been provided in Table 7 to show the effectiveness of the proposed method in the multi-class multiple type outlier detection problem.

5. Statistical comparison of the methods

We have performed a paired t-test (Table 8) to compare the performance of each classifier with the other with 95% confidence level (significance level $\alpha = 0.05$ for two tailed distribution). It can be seen that use of autoencoders have significantly improved the performance. For any two classifiers, let c_1^i and c_2^i be the respective performance scores (in our case, the value of ROC-AUC) on the i^{th} out of the N data sets (here $N = 8$). Then, the difference for the i^{th} dataset will be,

$$d_{12}^i = c_2^i - c_1^i. \tag{13}$$

Now, the average difference will be,

$$\bar{d}_{12} = \frac{1}{N}\sum_{i=1}^{N} d_{12}^i. \tag{14}$$

Now, the t-value for the two classifiers will be,

$$t_{12} = \frac{\bar{d}_{12}}{\sigma_{12}/\sqrt{N}}, \tag{15}$$

where, where σ_{12} is the standard deviation of the differences between the two classifiers over the N datasets.

This t-statistic will be distributed according to the t-distribution with $N - 1$ degrees of freedom (we have considered the one-tailed distribution in this case.) Since, we are comparing over 8 datasets (i.e. $N = 8$), our degrees of freedom will be $8 - 1 = 7$. (We have omitted the results for the multiple outlier type datasets since the method *i*Forests was replaced with Random Forests in that case.) The corresponding p-values are provided in Table 9. Taking the

Table 8
t-values for each pair of classifiers over the 8 single outlier type datasets.

	SAE-PNN	SAE-kNN	SAE-*i*Forests	PNN	kNN	*i*Forests
SAE-PNN		4.01	2.55	5.13	2.70	3.73
SAE-kNN			−2.63	−0.05	0.72	2.12
SAE-*i*Forests				2.25	2.37	3.56
PNN					0.41	1.93
kNN						2.08
*i*Forests						

Table 9
p-values for each pair of classifiers over the 8 single outlier type datasets.

	SAE-PNN	SAE-kNN	SAE-*i*Forests	PNN	kNN	*i*Forests
SAE-PNN		0.002553	0.019095	0.000679	0.015268	0.003677
SAE-kNN			0.016995	0.481958	0.246100	0.035730
SAE-*i*Forests				0.029731	0.024950	0.004629
PNN					0.347259	0.047567
kNN						0.037999
*i*Forests						

Table 10
Significance results for 95% confidence (here 1 means significant and 0 means non-significant).

	SAE-PNN	SAE-kNN	SAE-*i*Forests	PNN	kNN	*i*Forests
SAE-PNN		1	1	1	1	1
SAE-kNN			1	0	0	0
SAE-*i*Forests				0	0	1
PNN					0	0
kNN						0
*i*Forests						

Bonferroni [39] correction into consideration, we get the Bonferroni corrected critical *p*-values as

$$p_{critical}^{corrected} = 1 - (1 - p_{critical}/n_t)^{n_t}, \tag{16}$$

where, n_t denotes the number of comparison tests made. In our case, since fifteen comparisons were made, $n_t = 15$. If we set our critical value of p to $p_{critical} = 0.025$, (which corresponds to a 95% confidence interval for one-tailed tests) then the Bonferroni corrected critical value will be $p_{critical}^{corrected} = 0.025/15 = 0.024710429$.

If the p-value for the two classifiers p_{12} is greater than the critical value $p_{critical}^{corrected} = 0.024710429$, we reject the null hypothesis. We can see from Table 10 that the use of stacked autoencoders clearly enhanced the performance to a significant level. As both the methods *k*NN and PNN have identical hypotheses, the performance of a PNN classifier and a *k*NN classifier is comparable. Using SAE with iForests gave better results than using iForests alone. However, using stacked autoencoders with *k*NN did not give a significant improvement over *k*NN. Classification with *k*NN in the original feature space proved out to be comparable with the classification in the space of features extracted by SAE. Interestingly though, SAE-*k*NN could outperform SAE-iForest where *k*NN and iForests were comparable. Using an ensemble of PNNs in the feature space extracted by the SAE proved to be the best. The proposed method outperformed every other method. We can therefore say that non-linear transformations given by the autoencoders are helpful for highly imbalanced datasets for outlier detection.

6. Conclusion

In this article, we have proposed a supervised setting of outlier detection using deep learning in both single type outlier as well as multiple outlier type datasets. We have used stacked autoencoders as feature extractors and then have used a set of PNNs to classify the outliers and inliers in the data. Such a system is shown to perform better than the existing methods for both multi-class single type outlier detection as well as multi-class multiple type outlier detection problems. We can conclude that the use of stacked autoencoders clearly enhances the performance of all the methods. However, the proposed method outperforms all other methods. Experiments have also revealed that going deep (or using more hidden layers) in the feature extraction stage provided an enhancement of performance of the proposed method. But, too much depth degrades the performance again. In future, we would like to extend our work of multiple outlier type detection to unsupervised methods like SVDD, which currently put every new outlier to a single outlier class only.

Acknowledgement

We would like to thank Indian Statistical Institute for funding the research. We would also like to thank the reviewers for their detailed comments and suggestions.

References

[1] D. Hawkins, Identification of outliers, Springer Netherlands, 2014. https://books.google.co.in/books?id=toR6oAEACAAJ.
[2] A. Ghosh, Big data and its utility, Consult. Ahead 10 (1) (2016) 52–68.
[3] A. Amin, S. Anwar, A. Adnan, M. Nawaz, N. Howard, J. Qadir, A. Hawalah, A. Hussain, Comparing oversampling techniques to handle the class imbalance problem: a customer churn prediction case study, IEEE Access 4 (2016) 7940–7957.
[4] R. Domingues, M. Filippone, P. Michiardi, J. Zouaoui, A comparative evaluation of outlier detection algorithms: experiments and analyses, Pattern Recognit. 74 (2018) 406–421.
[5] G. Attarde, A. Deshpande, Outlier detection using unsupervised and semi-supervised technique on high dimensional data, Int. J. Innovative Res. Sci. Eng. Technol. 5 (7) (2016) 14180–14185.
[6] G.O. Campos, A. Zimek, J. Sander, R.J. Campello, B. Micenková, E. Schubert, I. Assent, M.E. Houle, On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study, Data Min. Knowl. Discov. 30 (4) (2016) 891–927.

[7] M. Radovanović, A. Nanopoulos, M. Ivanović, Reverse nearest neighbors in unsupervised distance-based outlier detection, IEEE Trans. Knowl. Data Eng. 27 (5) (2015) 1369–1382.
[8] K. Zhang, M. Hutter, H. Jin, A new local distance-based outlier detection approach for scattered real-world data, Adv. Knowl. Discovery Data Min. (2009) 813–822.
[9] L. Zhang, J. Lin, R. Karim, Adaptive kernel density-based anomaly detection for nonlinear systems, Knowl. Based Syst. 139 (2018) 50–63.
[10] C. Désir, S. Bernard, C. Petitjean, L. Heutte, One class random forests, Pattern Recognit. 46 (12) (2013) 3490–3506.
[11] D.M.J. Tax, R.P.W. Duin, Support vector data description, Mach. Learn. 54 (1) (2004) 45–66.
[12] Q. Liu, R. Klucik, C. Chen, G. Grant, D. Gallaher, Q. Lv, L. Shang, Unsupervised detection of contextual anomaly in remotely sensed data, Remote Sens. Environ. 202 (2017) 75–87.
[13] S.M. Erfani, S. Rajasegarar, S. Karunasekera, C. Leckie, High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning, Pattern Recognit. 58 (2016) 121–134.
[14] X. Wang, X.L. Wang, Y. Ma, D.M. Wilkes, A fast MST-inspired kNN-based outlier detection method, Inf. Syst. 48 (2015) 89–112.
[15] M.M. Breunig, H.P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, in: ACM Sigmod Record, 29, ACM, 2000, pp. 93–104.
[16] Y. Chen, D. Miao, H. Zhang, Neighborhood outlier detection, Expert Syst. Appl. 37 (12) (2010) 8745–8749.
[17] Z.G. Liu, Q. Pan, J. Dezert, A new belief-based k-nearest neighbor classification method, Pattern Recognit. 46 (3) (2013) 834–844.
[18] G. Bhattacharya, K. Ghosh, A.S. Chowdhury, kNN classification with an outlier informative distance measure, in: International Conference on Pattern Recognition and Machine Intelligence, Springer, 2017, pp. 21–27.
[19] M.S. Sadooghi, S.E. Khadem, Improving one class support vector machine novelty detection scheme using nonlinear features, Pattern Recognit. 83 (2018) 14–33.
[20] F.T. Liu, K.M. Ting, Z.H. Zhou, Isolation-based anomaly detection, ACM Trans. Knowl. Discovery Data (TKDD) 6 (1) (2012) 3.
[21] M.J. Kurian, R.S. Gladston, Improving the performance of a classification based outlier detection system using dimensionality reduction techniques, Int. J. Adv. Res. Comput. Sci. 8 (9) (2017) 4–9.
[22] A. Ghosh, N.R. Pal, S.K. Pal, Self-organization for object extraction using a multilayer neural network and fuzziness measures, IEEE Trans. Fuzzy Syst. 1 (1) (1993) 54.
[23] J. An, S. Cho, Variational autoencoder based anomaly detection using reconstruction probability, SNU Data Mining Center, Technical Report (2015).
[24] A.L. Oliveira, F.R. Costa, C.O. Filho, Novelty detection with constructive probabilistic neural networks, Neurocomputing 71 (4–6) (2008) 1046–1053.
[25] V. Jumutc, J.A.K. Suykens, Multi-class supervised novelty detection, IEEE Trans. Pattern Anal. Mach. Intell. 36 (12) (2014) 2510–2523.
[26] A. Mellor, S. Boukir, A. Haywood, S. Jones, Exploring issues of training data imbalance and mislabelling on random forest performance for large area land cover classification using the ensemble margin, ISPRS J. Photogramm. Remote Sens. 105 (2015) 155–168.
[27] Y. Wang, H. Yao, S. Zhao, Auto-encoder based dimensionality reduction, Neurocomputing 184 (2016) 232–242.
[28] A. Romero, C. Gatta, G. Camps-Valls, Unsupervised deep feature extraction for remote sensing image classification, IEEE Trans. Geosci. Remote Sens. 54 (3) (2016) 1349–1362.
[29] C.C. Olson, K.P. Judd, J.M. Nichols, Manifold learning techniques for unsupervised anomaly detection, Expert Syst. Appl. 91 (2018) 374–385.
[30] H.C. Shin, M.R. Orton, D.J. Collins, S.J. Doran, M.O. Leach, Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4d patient data, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1930–1943.
[31] S. Sathe, C.C. Aggarwal, LODES: local density meets spectral outlier detection, in: Proceedings of the 2016 SIAM International Conference on Data Mining, SIAM, 2016, pp. 171–179.
[32] C.C. Aggarwal, S. Sathe, Theoretical foundations and algorithms for outlier ensembles, ACM SIGKDD Explor. Newsl. 17 (1) (2015) 24–47.
[33] F.T. Liu, K.M. Ting, Z.H. Zhou, Isolation forest, in: Eighth IEEE International Conference on Data Mining (ICDM), 2008., IEEE, 2008, pp. 413–422.
[34] T.R. Bandaragoda, K.M. Ting, D. Albrecht, F.T. Liu, J.R. Wells, Efficient anomaly detection by isolation using nearest neighbour ensemble, in: IEEE International Conference on Data Mining Workshop (ICDMW), 2014, IEEE, 2014, pp. 698–705.
[35] R. Lyon, B. Stappers, S. Cooper, J. Brooke, J. Knowles, Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach, Mon. Not. R. Astron. Soc. 459 (1) (2016) 1104–1123.
[36] A. Dal Pozzolo, O. Caelen, R.A. Johnson, G. Bontempi, Calibrating probability with undersampling for unbalanced classification, in: IEEE Symposium Series on Computational Intelligence, 2015, IEEE, 2015, pp. 159–166.
[37] B. Micenková, B. McWilliams, I. Assent, Learning outlier ensembles: the best of both worlds - supervised and unsupervised, in: Proceedings of the ACM SIGKDD 2014 Workshop on Outlier Detection and Description under Data Diversity (ODD2). New York, NY, USA, 2014, pp. 51–54.
[38] P.R. Nicolas, Scala for machine learning: data processing, ML algorithms, smart analytics, and more, Packt Publishing, 2017. https://books.google.co.in/books?id=9plGDwAAQBAJ.
[39] J.M. Bland, D.G. Altman, Multiple significance tests: the Bonferroni method, BMJ 310 (6973) (1995) 170.

Debasrita Chakraborty is a Senior Research Fellow at Machine Intelligence Unit, Indian Statistical Institute. She received the B.Sc. (Honours) degree in physics from Vivekananda Mahavidyalaya, Burdwan, India, in 2012, and the M.Sc. degree in physics from the University of Burdwan, Burdwan, India, in 2014. Her current research interests include Deep Neural Networks, Data Science, Machine Learning and Pattern Recognition.

Vaasudev Narayanan received his B. Tech (Honours) degree in computer science from Indian Institute of Technology (Indian School of Mines), Dhanbad, India in 2018. He will be joining the Centre for Visual Information Technology at International Institute of Information Technology, Hyderabad, India as a Research Fellow. His current research interests include Neural Networks and Machine Learning.

Ashish Ghosh is a professor at the Machine Intelligence Unit, Indian Statistical Institute. He has already published more than 170 research papers in internationally reputed journals and refereed conferences, and has edited eight books. His current research interests include Pattern Recognition and Machine Learning, Deep Learning, Data Mining, Big Data Analysis, Image Analysis, Remotely Sensed Image Analysis, Video Image Analysis, Soft Computing, Fuzzy Sets and Uncertainty Analysis, Neural Networks, Evolutionary Computation, and Bioinformatics. Dr. Ghosh received the prestigious and most coveted Young Scientists Award in Engineering Sciences from the Indian National Science Academy in 1995, and in Computer Science from the Indian Science Congress Association in 1992. He was selected as an Associate of the Indian Academy of Sciences, Bangalore, India, in 1997. He is a member of the founding team that established the National Center for Soft Computing Research at the Indian Statistical Institute, Kolkata, in 2004, with funding from the Department of Science and Technology, Government of India, and is currently the In-charge of the Center. He is acting as a member of the editorial boards of various international journals.