LUNAR: Unifying Local Outlier Detection Methods via Graph Neural Networks

Adam Goodge[1,3]**, Bryan Hooi** [1,2]**, See-Kiong Ng** [1,2]**, Wee Siong Ng** [3]

[1] School of Computing, National University of Singapore
[2] Institute of Data Science, National University of Singapore
[3] Institute for Infocomm Research, A*STAR, Singapore
adam.goodge@u.nus.edu, {dcsbhk, seekiong}@nus.edu.sg, wsng@i2r.a-star.edu.sg

Abstract

Many well-established anomaly detection methods use the distance of a sample to those in its local neighbourhood: so-called 'local outlier methods', such as LOF and DBSCAN. They are popular for their simple principles and strong performance on unstructured, feature-based data that is commonplace in many practical applications. However, they cannot learn to adapt for a particular set of data due to their lack of trainable parameters. In this paper, we begin by unifying local outlier methods by showing that they are particular cases of the more general message passing framework used in graph neural networks. This allows us to introduce learnability into local outlier methods, in the form of a neural network, for greater flexibility and expressivity: specifically, we propose LUNAR, a novel, graph neural network-based anomaly detection method. LUNAR learns to use information from the nearest neighbours of each node in a trainable way to find anomalies. We show that our method performs significantly better than existing local outlier methods, as well as state-of-the-art deep baselines. We also show that the performance of our method is much more robust to different settings of the local neighbourhood size.

Introduction

Unsupervised anomaly detection is the task of detecting anomalies within a set of data without relying on ground truth labels of known anomalies. It is an extremely important task in a wide range of practical applications and has therefore received a great amount of research interest. As anomalies tend to be much rarer than normal data, labelled anomalies are difficult to obtain in the quantity needed to adequately train supervised techniques.

Many well-established unsupervised methods detect anomalies by measuring the distance of a point to its nearest neighbouring points: so-called **local outlier methods**, such as LOF and DBSCAN. These methods are very popular in practice due to their straightforward principles and assumptions, as well as their interpretable outputs. In our experiments, we also find that their performance also holds up favourably against more recent, deep learning-based methods. The latter have to fully embed knowledge about normal and abnormal regions of the data space in their network

parameters. They are mostly designed for highly structured, high-dimensional data such as images, but their performance often struggles for the less structured, feature-based data that is commonly used in many applications. As such, local outlier methods remain the default choice is many areas.

A range of local outlier methods have been developed, each with its own unique formulation and properties. However, many of them also share common characteristics with each other. In this paper, our first contribution is to unify local outlier methods under a simple, general framework based on the message passing scheme used in **graph neural networks** (GNN). We demonstrate that many popular methods, such as KNN, LOF and DBSCAN, can be seen as particular cases of this more general message passing framework.

Despite their popularity, local outlier methods lack the capacity to learn to optimise for or adapt to a particular set of data, e.g. through trainable parameters. Furthermore, in an unsupervised setting, there is no straightforward way to find optimal hyper-parameter settings, such as the number of nearest neighbours, which is extremely important and greatly affects performance. In this paper, we also propose a novel method named LUNAR (**L**earnable **U**nified **N**eighbourhood-based **A**nomaly **R**anking), which is based on the same message passing framework for local outlier methods but addresses their shortcomings by enabling **learnability** via graph neural networks.

In summary, we make the following contributions:

- We show that many popular local outlier methods, such as KNN, LOF and DBSCAN, can be unified under a single framework based on graph neural networks.

- We use this framework to develop a novel, GNN-based anomaly detection method (LUNAR) which is more flexible and adaptive to a given set of data than local outlier methods due to its trainable parameters.

- We show that our method gives better performance[1] than popular classical methods, including local outlier methods, as well as state-of-the-art deep learning-based methods in anomaly detection. We also show that its performance is much more robust to different settings of the local neighbourhood size than local outlier methods.

---

[1]Code available at https://github.com/agoodge/LUNAR

Related Work

Neighbourhood-based Anomaly Detection Besides notable examples like OC-SVM (Schölkopf et al. 2001) and IFOREST (Liu, Ting, and Zhou 2008), most classical anomaly detection methods directly measure the distance of a point to its nearest neighbours to detect anomalies, which we call 'local outlier methods'. They rely on the assumption that anomalies are in sparse regions of the data space, far away from highly dense clusters of normal points. Points that are close to their neighbours are more likely to be normal themselves, whilst points far from their neighbours are more likely to be anomalies.

KNN (Angiulli and Pizzuti 2002) uses the distance to the kth nearest neighbour as the anomaly score. Alternatively, DBSCAN (Ester et al. 1996), which simultaneously learns to to cluster normal data while also detecting outliers, uses the number of points within a pre-defined distance.

Local Outlier Factor (LOF) (Breunig et al. 2000) measures distances to define a density measure and compares this density to neighbouring points. Various extensions and variants of have been developed, including but not limited to: Local Outlier Probabilities (LoOP) (Kriegel et al. 2009a), Connectivity-based (COF) (Tang et al. 2002), Local Correlation Integral (LOCI) (Papadimitriou et al. 2003), Influenced Outlierness (INFLO) (Jin et al. 2006), and Subspace Outlier Detection (SOD) (Kriegel et al. 2009b).

These methods suffer from a lack of learnability: they do not use the information in the training set to optimise model parameters for better anomaly scoring. Instead, they are based on pre-defined heuristics and hyper-parameters. These settings strongly influence performance, yet the optimal settings are very difficult to validate before deployment without access to labelled anomalies.

Deep Learning-based Anomaly Detection Deep models have improved state-of-the-art performance in anomaly detection for highly structured, high dimensional data especially. Autoencoders are particularly popular, with the reconstruction error acting as the anomaly score. Normal samples are assumed to be reconstructed with lower error than anomalies. They have been used with fully-connected (Sakurada and Yairi 2014), convolutional (Zhao et al. 2017) or recurrent (Malhotra et al. 2015) layers for different data applications. Variational (An 2015), denoising (Feng and Han 2015) and adversarial (Vu et al. 2019) autoencoders have also been used. The reconstruction errors from each encoder-decoder layer pair are fused together in (Kim et al. 2019). The autoencoder latent encodings are optimised directly in (Goodge et al. 2020) to improve robustness against adversarial perturbations.

Others use deep models as feature extractors for a secondary anomaly-detecting module, such as KNN (Bergman, Cohen, and Hoshen 2020), KDE (Nicolau, McDermott et al. 2016), DBSCAN (Amarbayasgalan, Jargalsaikhan, and Ryu 2018) or autoregressive models (Abati et al. 2019). Zong et al. (2018) simultaneously train an autoencoder for feature extraction with a Gaussian mixture model in the latent space for anomaly detection. Ruff et al. (2018) learn a normality-encoding hypersphere in the latent space and the anomaly score is the distance from the centre. Generative adversarial networks use the ability of the generator to generate an unseen sample to indicate its anomalousness (Schlegl et al. 2017; Zenati et al. 2018).

There has been some interest in GNNs for anomaly detection in graph data, such as sensor networks (Deng and Hooi 2021; Cai et al. 2020; Zheng et al. 2019). Our method also uses GNNs, though it is distinct from these works as it is designed for unstructured, feature-based data rather than graphs.

Background

Local Outlier Methods

'Local outlier methods' refers to those methods which directly use the distance of a point to its k nearest neighbours to determine its anomalousness. We now detail KNN and LOF.

KNN The anomaly score of a point $\mathbf{x}_i$ is its distance to its kth nearest neighbour:

$$\text{KNN}(\mathbf{x}_i) = \text{dist}(\mathbf{x}_i, \mathbf{x}_i^{(k)}) \qquad (1)$$

where $\mathbf{x}_i^{(k)}$ is the kth nearest neighbour of $\mathbf{x}_i$. Euclidean distance is most common, though any distance measure could be used depending on its suitability to the data type.

LOF The Local Outlier Factor instead uses the 'reachability distance', which is defined for $\mathbf{x}_i$ from $\mathbf{x}_j$ as:

$$\text{reach}_k(\mathbf{x}_i, \mathbf{x}_j) = \max\{\text{k-dist}(\mathbf{x}_j), \text{dist}(\mathbf{x}_i, \mathbf{x}_j)\} \quad (2)$$

where $\text{k-dist}(\mathbf{x}_j)$ is equal to $\text{dist}(\mathbf{x}_j, \mathbf{x}_j^{(k)})$. This is used to calculate the 'local reachability density' of a point:

$$\text{lrd}_k(A) := \left(\frac{\sum\limits_{j \in \mathcal{N}_i} \text{reach}_k(\mathbf{x}_i, \mathbf{x}_j)}{|\mathcal{N}_i|} \right)^{-1} \qquad (3)$$

where $\mathcal{N}_i$ is the set of k nearest neighbours of $\mathbf{x}_i$. Finally, this density measure is compared with that of neighbouring points to determine the local outlier factor:

$$\text{LOF}(\mathbf{x}_i) = \frac{\sum_{j \in \mathcal{N}_i} \text{lrd}_k(\mathbf{x}_j)}{|\mathcal{N}_i| \cdot \text{lrd}_k(\mathbf{x}_i)}. \qquad (4)$$

Graph Neural Networks

Graph neural networks (GNN) operate on a graph $G(V, E)$, in which a node, $i \in V$, is connected to an adjacent node, $j \in V$, via an edge $(j, i) \in E$. Edges can be undirected, in which case information flows in both directions between adjacent nodes. Alternatively, if the edges are directed, then information only flows from the source node to target node, i.e. from j to i along the edge (j, i). Nodes and edges can, but need not, have feature vectors, denoted by $\mathbf{x}_i$ and $e_{j,i}$ for node i and edge (j, i) respectively.

GNNs have become increasingly popular in a range of graph-related applications, such as social networks (Fan

et al. 2019) and traffic networks (Cui et al. 2019). Of particular interest here is the node classification task, which involves learning successive latent representations of nodes through the network layers in order to predict the class label of each node. This relies on a message passing scheme, made up of *message*, *aggregation* and *update* steps. The message function (ϕ) determines the information to be sent to the node in question from each neighbour. The aggregation function ($\square$) summarises these incoming messages into one message, for example by average or max-pooling. Finally, the update function (γ) uses this aggregated message and the current representation of the node to compute its subsequent representation. In summary, the kth layer of a GNN calculates the hidden representation of a node via the following (Gilmer et al. 2017):

$$\mathbf{h}_{\mathcal{N}_i}^{(k)} = \underset{j \in \mathcal{N}_i}{\square} \phi^{(k)}(\mathbf{h}_i^{(k-1)}, \mathbf{h}_j^{(k-1)}, \mathbf{e}_{j,i}),$$

$$\mathbf{h}_i^{(k)} = \gamma^{(k)}(\mathbf{h}_i^{(k-1)}, \mathbf{h}_{\mathcal{N}_i}^{(k)}). \tag{5}$$

where $h_i^{(0)} = \mathbf{x}_i$ and $\mathcal{N}_i$ is the set of adjacent nodes to i. $\mathbf{h}_{\mathcal{N}_i}^{(k)}$ is the aggregation of the messages from its neighbours.

Problem Definition

We now define the unsupervised anomaly detection problem of interest in this paper. We assume to have m normal training samples $\mathbf{x}_1^{(\text{train})}, ..., \mathbf{x}_m^{(\text{train})} \in \mathbb{R}^d$ and n testing samples, $\mathbf{x}_1^{(\text{test})}, ..., \mathbf{x}_n^{(\text{test})} \in \mathbb{R}^d$, each of which may be normal or anomalous. For a test sample $\mathbf{x}_i^{(\text{test})}$, our algorithm should output an **anomaly score** $s(\mathbf{x}_i^{(\text{test})})$ that is low (or high) if $\mathbf{x}_i^{(\text{test})}$ is normal (or anomalous).

In local outlier methods, the fundamental question is:

How should the distances of a sample $\mathbf{x}_i^{(\text{test})}$ to its nearest neighbours be used in computing its anomaly score?

In the following section, we show that many local outlier methods can be seen as particular cases of the message passing framework used by GNNs.

Unifying Framework

Local outlier methods collect information from the nearest neighbouring points to compute a statistic to indicate the anomalousness of a given point. This process fits within the GNN message passing framework outlined in (5). For ease of understanding, we show this using the example of KNN in particular.

Example: KNN

Recall that KNN computes the anomaly score based on the distance to the kth nearest neighbour of a point.

In the context of message passing, each data sample corresponds to one node in a graph and node i is connected to each of its k nearest neighbours, $j \in \mathcal{N}_i$, via a directed edge (j, i), with edge feature $\mathbf{e}_{j,i}$ equal to the distance between them (k-NN graph):

$$e_{j,i} = \begin{cases} \mathsf{dist}(\mathbf{x}_i, \mathbf{x}_j) \text{ if } j \in \mathcal{N}_i. \\ 0 \text{ otherwise.} \end{cases} \tag{6}$$

Step	KNN	LOF	DBSCAN
$\mathbf{e}_{j,i}$	$\mathsf{dist}(\mathbf{x}_i, \mathbf{x}_j)$	$\mathsf{reach}(\mathbf{x}_i, \mathbf{x}_j)$	$\mathsf{dist}(\mathbf{x}_i, \mathbf{x}_j)$
$\phi^{(1)}$	$\mathbf{e}_{j,i}$	$\mathbf{e}_{j,i}$	$H(\epsilon - \mathbf{e}_{j,i})$
$\square^{(1)}$	max	sum	sum
$\gamma^{(1)}$	$\mathbf{h}_{\mathcal{N}_i}^{(1)}$	$1/\mathbf{h}_{\mathcal{N}_i}^{(1)}$	$H(\mathbf{h}_{\mathcal{N}_i}^{(1)} - \text{minPts})$
$\phi^{(2)}$	-	$\mathbf{h}_j^{(1)}/\mathbf{h}_i^{(1)}$	$\mathbf{h}_j^{(1)}$
$\square^{(2)}$	-	mean	max
$\gamma^{(2)}$	-	$\mathbf{h}_{\mathcal{N}_i}^{(2)}$	$1 - \mathbf{h}_{\mathcal{N}_i}^{(2)}$

Table 1: Local outlier methods as they relate to the message passing framework defined in (5). H refers to the Heaviside function.

These edges are directed as $j \in \mathcal{N}_i \implies i \in \mathcal{N}_j$, so information flows along edge (j, i) only from the source node j to target node i. With this graph, we now show that KNN can be explained in terms of the *message*, *aggregation* and *update* functions in (5).

Message KNN collects the distances of a node to its nearest neighbours:

$$\phi^{(1)} := \mathbf{e}_{j,i}. \tag{7}$$

Aggregation It then outputs the maximum of these distances (i.e. max-pooling):

$$\mathbf{h}_{\mathcal{N}_i}^{(1)} := \max_{j \in \mathcal{N}_i} \phi^{(1)} \tag{8}$$

Update Finally, it outputs this aggregated message as the anomaly score:

$$\gamma^{(1)} := \mathbf{h}_{\mathcal{N}_i}^{(1)} \tag{9}$$

Proposition 1. KNN *is a special case of the message passing scheme formulated in (5).*

Proof. The KNN anomaly score can be calculated using the *message*, *aggregation* and *update* functions formulated above. By substituting these functions into their appropriate counterparts in (5), we arrive at the following:

$$\mathrm{KNN}(i) = \max_{j \in \mathcal{N}_i}(\mathbf{e}_{j,i}), \tag{10}$$

which is a special, one-layer case of the message passing framework in (5). □

A similar analysis can be applied to LOF and DBSCAN, which are instead two-layer cases with two rounds of message passing. For example, in LOF, the first layer calculates the local reachability density as in (3) and the second layer calculates the local outlier score as in (4). Table 1 formalizes these connections, and an extended version with more local outlier methods can be found in the supplementary material available online[1].
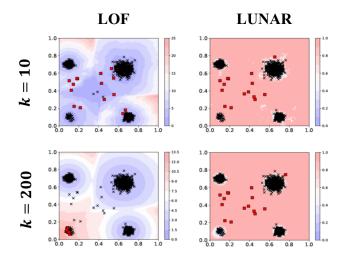
LOF **LUNAR**

$k = 10$

$k = 200$

Figure 1: Contours of the scores assigned by LOF versus LUNAR. Red indicates a high (anomalous) score and blue indicates a low (normal) score. Points are marked by black crosses and those with the top 15 highest assigned scores are marked by red squares.

Motivation: The Importance of Learnability

Local outlier methods lack trainable parameters which enable them to optimise their performance for a given training set. In this section, we show that this hinders their overall accuracy. To do this, we compare the performance of LOF against our novel methodology LUNAR, on a toy training dataset of 1000 points sampled from four Gaussian distributions. As perfectly pure training sets are rare in practice, we also generate 15 points from a uniform distribution within the data bounds. These points are much rarer and sparser than the others, so they should not significantly influence the predicted normal regions.

In Figure 1, low scores (blue) indicate a predicted normal region while high scores (red) indicate a predicted anomalous region. The points with one of the top 15 anomaly scores are indicated with red squares. We test the methods with a small and large value for the hyperparameter dictating the number of nearest neighbours (k).

With low k, the LOF score is low around the four clusters, but also low far away from these clusters with little or no nearby training points. The central outlying region especially appears normal due to the strong influence of the relative sparsity of the very few points in the area. Conversely, with large k, the LOF score is erroneously high for the smaller cluster in the bottom-left corner. LOF fails to recognise the clusters existence as it contains fewer points than k, instead predicting all nearby points to be anomalous. These issues are challenging as local outlier methods lack the capacity to learn a more optimal scoring mechanism from the data directly.

In comparison, the learnability of LUNAR enables it to perform better and more robustly across k: the regions assigned with normal or anomalous scores are a much closer fit to the training data, and the highest anomaly scores are given to the sparse, central points more accurately. We now describe its methodology in full.

LUNAR: Methodology

Overview Our methodology involves a one layer graph neural network as per the message passing framework described by (5). We represent a set of data as a graph, with a node corresponding to each data sample and directed edges connecting a target node to a set of source nodes, which are the nearest neighbours of the samples. For a given target node, the network utilises information from its its nearest neighbouring nodes to learn its anomaly score. It differs from other GNN implementations for several reasons:

- We construct the k-NN graph of any feature-based, tabular dataset, rather than being restricted to graph datasets.
- We use a node's distances to its k nearest neighbours as input, which is more generalizable than using its feature vector.
- We use a learnable message aggregation function, whereas most GNNs use a fixed aggregation approach.

Model Design

We now describe the methodology used in LUNAR in more detail, starting with how the graph is formulated.

Nearest Neighbourhood Graph For a data sample $\mathbf{x}_i$, we define a target node i and edge (j, i) connecting it to a source node j for all j where $\mathbf{x}_j$ is in the set of k nearest neighbours to $\mathbf{x}_i$. The edge feature vector is equal to the Euclidean distance between the two points:

$$\mathbf{e}_{j,i} = \begin{cases} \mathsf{dist}(\mathbf{x}_i, \mathbf{x}_j) \text{ if } j \in \mathcal{N}_i. \\ 0 \text{ otherwise.} \end{cases} \quad (11)$$

As training samples are all assumed to be normal, we only search for nearest neighbours among training samples, so that anomalies cannot influence the neighbourhood. With this, we define the *message*, *aggregation* and *update* functions in (5) as follows:

Message The message passed from source node j to target node i along edge (j, i) is equal to the edge feature $e_{j,i}$ (i.e. the distance between the points):

$$\phi^{(1)} := \mathbf{e}_{j,i}. \quad (12)$$

Aggregation Rather than a fixed average or max-pooling, we use a learnable aggregation, which is suitable for our setting as we are dealing with node neighbourhoods of a fixed size (k). Our message aggregation involves concatenating them to give a k-dimensional vector, $\mathbf{e}^{(i)}$, where each entry represents the distance of $\mathbf{x}_i$ to its corresponding neighbour:

$$\mathbf{e}^{(i)} := [\mathbf{e}_{1,i}, ..., \mathbf{e}_{k,i}] \in \mathbb{R}^k. \quad (13)$$

This vector is mapped to a single, scalar value representing the anomalousness of node i, through a neural network:

$$\mathbf{h}_{\mathcal{N}_i}^{(1)} := \mathcal{F}(\mathbf{e}^{(i)}, \Theta), \quad (14)$$

where Θ are the weights of the neural network $\mathcal{F}$.

Dataset	#Size	#Dim	#Anomalies
HRSS	90515	20	10187
MI-F	24955	58	2050
MI-V	22905	58	3942
OPTDIGITS	5216	64	150
PENDIGITS	6870	16	156
SATELLITE	6435	36	399
SHUTTLE	49097	9	3511
THYROID	7200	21	534

Table 2: Statistics of the datasets used in experiments.

Update   Finally, the update function outputs this learned, aggregated message:

$$\gamma^{(1)} := \mathbf{h}_{\mathcal{N}_i}^{(1)}. \tag{15}$$

We use a loss function which trains the GNN to output a score of 0 for normal nodes and 1 for anomalous nodes. As all training points are of the normal class, the network would attain perfect training accuracy by outputting zero scores regardless of the input. To avoid this trivial solution, we generate negative samples to act as artificial anomalies, training the model to output a score of 1 for the negative sample nodes. With this, we aim to learn a decision boundary between normal samples and negative samples which generalises to the true anomalies in the test set. In the next section, we detail how negative samples are generated.

Negative Sampling

Negative samples have been used to introduce supervision to unsupervised tasks, such as in contrastive learning (Chen et al. 2020), as well as anomaly detection (Sipple 2020). They need to be sufficiently distinguishable from normal samples for the model to learn the decision boundary, but not so dissimilar that the task is too easy and the learnt boundary fails to discriminate normal samples from real anomalies. With this in mind, we combine two methods of generating negative samples, which are as follows:

Uniform   The first method involves generating negative samples from a uniform distribution:

$$\mathbf{x}^{(\text{negative})} \sim \mathcal{U}(-\varepsilon, 1 + \varepsilon) \in \mathbb{R}^d, \tag{16}$$

where ε is a small, positive constant. For simplicity, we use $\epsilon = 0.1$ in all experiments. The training data is normalized to the range $[0, 1]$, so these samples cover the data bounds. However, normal data occupies a much smaller subspace within these bounds, so many of these negative samples would be far from normal data and ineffective for learning the decision boundary. We complements this by generating an additional set of more 'difficult' negative samples.

Subspace Perturbation   In the second method, we generate negative samples by adding Gaussian noise to normal samples in a subset of their feature dimensions:

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, I) \in \mathbb{R}^d,$$
$$\mathbf{x}_i^{(\text{negative})} = \mathbf{x}_i^{(\text{train})} + \mathbf{M} \circ \varepsilon \mathbf{z}. \tag{17}$$

where ε is a small, positive constant and $\mathbf{M} \in \mathbb{R}^d$ is a vector of binary random variables. Each element in $\mathbf{M}$ has probability p of being one (and $1 - p$ of being zero), which determines the feature dimensions to be perturbed. We use $p = 0.3$ in all experiments.

Computational Runtime   In the supplementary material, we show the runtimes of LUNAR versus other methods in experiments. We see that LUNAR is faster than the other deep methods tested (e.g. 33.71 seconds for LUNAR versus 55.92 seconds for DAGMM on the HRSS dataset). LUNAR avoids directly training on high-dimensional feature data in its input, instead using distances between points, which explains the faster training time.

Limitations   A limitation of LUNAR, as with all local outlier methods, is in finding the k nearest neighbours. This is mostly an issue in very high-dimensional spaces, such as with image data, where distance measures become less meaningful (Beyer et al. 1999). Adapting LUNAR for higher dimensionality is left for future work at present.

Theoretical Properties   An additional benefit of our unified approach is that we can use it to characterize theoretical properties of almost all local outlier methods in our framework (including LUNAR, KNN, LOF, and DBSCAN) in a unified way. One simple but important property of algorithms is their *symmetries under transformations*, which are very relevant to understanding their *inductive biases*, or the assumptions they use to generalize to unseen data.

Let $s(\mathbf{x}; \{\mathbf{x}_i^{(\text{train})}\}_{i=1}^m)$ be the anomaly score of any local outlier method evaluated at $\mathbf{x}$ given training data $\{\mathbf{x}_i^{(\text{train})}\}_{i=1}^m$.

Proposition 2 (Transformation Equivariance)*. Given any distance-preserving transformation f, the score s is transformation equivariant; that is,*

$$s(\mathbf{x}; \{\mathbf{x}_i^{(\text{train})}\}_{i=1}^m) = s(f(\mathbf{x}); \{f(\mathbf{x}_i^{(\text{train})})\}_{i=1}^m) \tag{18}$$

For example, s is equivariant to rotations, translations and reflections.

Proof. As shown in Table 1, all these methods compute distances $\text{dist}(\mathbf{x}_i, \mathbf{x}_j)$ or $\text{reach}(\mathbf{x}_i, \mathbf{x}_j)$ as input, and do not use the input features $\mathbf{x}_i$ in any other way. Applying f to the training and test data does not change the (reachability) distances between them, thus also preserving the score s. $\square$

Experiments

We now conduct experiments with real datasets to answer the following research questions:
RQ1 (Accuracy): Does LUNAR outperform existing baselines in detecting true anomalies?
RQ2 (Robustness): Is LUNAR more robust to changes in the neighbourhood size, k, than existing local outlier

Dataset	IFOREST	OC-SVM	LOF	KNN	AE	VAE	DAGMM	SO-GAAL	DN2	LUNAR
HRSS	59.61	61.03	60.13	62.09	61.16	63.30	55.93	45.90	60.20	**92.17****
MI-F	84.24	78.65	63.07	78.08	71.53	78.63	81.45	32.07	77.26	**84.37**
MI-V	84.28	74.56	79.14	82.71	82.42	75.96	78.19	55.34	62.54	**96.73****
OPTDIGITS	79.34	59.84	99.53	96.57	97.46	86.71	75.56	74.35	34.98	**99.76**
PENDIGITS	96.70	94.08	98.18	98.42	96.42	94.76	95.98	94.65	85.30	**99.81****
SATELLITE	80.10	64.64	84.25	**86.07**	81.48	66.09	78.22	84.16	75.37	85.35
SHUTTLE	99.64	98.29	99.80	99.56	99.26	98.33	99.51	99.38	96.97	**99.97****
THYROID	76.30	52.81	68.67	63.01	64.34	51.54	70.91	60.13	58.09	**85.44****

Table 3: AUC Score for each method on each dataset. Best scores are highlighted in bold. Average scores marked by ** are greater than the next best performing method with significance level $p < 0.01$, according to the t-test. More significance test results are found in the supplementary material.

methods?

RQ3 (Ablation Study): How do variations in our methodology affected its performance?

Datasets

Each dataset used in our experiments is publicly available and consists of a normal (0) class and anomaly (1) class. Table 2 summarises them and their key statistics.

As we focus on the unsupervised case, in which the training set only consists of samples labelled as normal (all labelled anomalies are in the test set). We use Area-Under-Curve (AUC) to measure performance. The relative proportion of anomalies in the test set does not affect the scoring of any individual point, so we can choose to randomly subsample normal points to achieve a 50:50 normal:anomaly ratio in the test set. Of the remaining normal samples, they are split 85:15 into a training set and validation set. We randomly generate both 'Uniform' and 'Subspace Perturbation' negative samples for the training and validation sets separately to avoid leaking information. We use a 1:1 ratio of negative:normal samples in both sets for all experiments.

Training Procedure

The neural network, $\mathcal{F}$ in (14), consists of four fully connected hidden layers all of size 256. All layers used tanh activation except for the sigmoid function at the output layer. We used mean squared error as the loss function and Adam (Kingma and Ba 2014) for optimization with a learning rate of 0.001 and weight decay of 0.1.

We trained the model for 200 epochs and used the model parameters with the best validation score as the final model. It was implemented using PyTorch Geometric on Windows OS and a Nvidia GeForce RTX 2080 Ti GPU.

Baselines

We use the PyOD library (Zhao, Nasrullah, and Li 2019) implementations of IFOREST, OC-SVM, LOF, KNN, and the GAN-based SO-GAAL (Liu et al. 2019). We also implement a deep autoencoder (AE) and VAE built in Pytorch, and DAGMM as in (Zong et al. 2018) with publicly available codes. Finally, DN2 (Bergman, Cohen, and Hoshen 2020), which performs KNN with latent features learnt from

a deep, pre-trained feature extractor. As we are interested in tabular data rather than image data, unlike the original paper, we use an autoencoder (the same model as in AE) for feature extraction.

RQ1 (Accuracy):

Table 3 shows the AUC score (multiplied by 100) of each method for each dataset. We use AUC as it does not rely on a user-defined score threshold to predict normal or anomalous labels. The scores shown are the average over five repeated trials with different random seeds. For the methods that use it, all results are with $k = 100$ as the number of neighbours unless stated otherwise.

We see that LUNAR gives the best performance on all datasets except SATELLITE, for which KNN is slightly better. For the HRSS, MI-V and THYROID datasets in particular, our method performs substantially better than the baselines: between 10 and 30 percentage points better than the second best method. Our scores marked by ** are significantly better than the second best performing method for each dataset with significance value $p < 0.01$ according to the t-test.

RQ2 (Robustness to Neighbourhood Size):

LOF, KNN and DN2 also use the k nearest neighbours of a point to determine its anomalousness. In Table 4, we show the performance of these methods for various k. We see that these methods depend greatly on the value of k. For example, their score decreases by 26, 24 and 25 percentage points respectively for HRSS as k increases from 2 to 200. In stark contrast, LUNAR only drops in performance by 3 points in the same range. LUNAR gives the best performance in the vast majority of datasets and k settings. Our method not only performs better, but maintains stronger performance for different settings of k. This is because it is able to learn to use the information from all k neighbours effectively, whereas the other methods lose information from most neighbours, as decided by a pre-set aggregation rule.

RQ3 (Ablation Study):

Table 5 shows the performance with Subspace Perturbation (SP) and Uniform (U) negative samples individually. SP negative samples give better performance than U samples

k	LOF	KNN	DN2	LUNAR	LOF	KNN	DN2	LUNAR	LOF	KNN	DN2	LUNAR
		HRSS				MI-F				MI-V		
2	82.08	86.25	85.28	**93.88**	90.43	77.84	**91.13**	81.50	94.31	94.58	86.76	**96.06**
10	67.98	65.53	62.40	**92.67**	**86.41**	73.46	85.58	82.39	92.60	88.53	77.92	**96.09**
50	61.66	62.71	60.46	**92.21**	67.17	71.69	78.66	**83.58**	78.61	83.29	64.96	**96.38**
100	60.13	62.09	60.20	**92.17**	63.07	78.08	77.26	**84.37**	79.14	82.71	62.54	**96.73**
150	57.22	61.81	60.14	**91.61**	60.60	80.79	76.33	**82.82**	80.73	82.86	61.77	**96.53**
200	55.59	61.86	60.22	**90.09**	70.89	82.85	75.93	**84.47**	81.75	82.65	61.67	**96.30**
Avg.	64.11	67.10	64.79	**92.11**	73.10	77.45	80.82	**83.19**	84.52	85.77	69.27	**96.35**
		OPTDIGITS				PENDIGITS				SATELLITE		
2	99.58	**99.91**	50.90	**99.91**	99.37	**99.84**	81.08	**99.84**	85.05	87.72	80.16	**87.80**
10	**99.92**	99.63	45.84	99.79	99.67	99.77	80.74	**99.82**	85.38	86.77	79.43	**87.83**
50	99.72	98.41	39.23	**99.81**	98.79	98.79	81.83	**99.80**	83.44	86.07	76.52	**87.58**
100	99.53	96.57	34.98	**99.76**	98.18	98.42	85.30	**99.81**	84.25	**86.07**	75.37	85.35
150	99.11	94.85	33.10	**99.73**	97.58	98.07	86.39	**99.76**	84.86	**85.85**	74.48	83.95
200	98.63	93.13	32.14	**99.78**	97.19	97.52	85.49	**99.71**	85.21	**85.46**	73.39	84.70
Avg.	99.41	97.09	39.37	**99.79**	98.46	98.74	83.47	**99.79**	84.69	**86.32**	76.55	86.08

k	LOF	KNN	DN2	LUNAR	LOF	KNN	DN2	LUNAR
		SHUTTLE				THYROID		
2	99.64	**99.98**	98.94	**99.98**	**83.70**	80.28	64.09	83.38
10	99.91	99.93	98.22	**99.95**	83.69	73.87	62.71	**84.24**
50	99.74	99.68	97.19	**99.97**	74.41	66.49	59.88	**86.01**
100	99.80	99.56	96.97	**99.97**	68.67	63.01	58.09	**85.44**
150	99.80	99.43	96.68	**99.95**	67.20	62.26	56.86	**86.08**
200	99.69	99.32	96.45	**99.97**	66.58	61.24	56.26	**86.67**
Avg.	99.76	99.65	97.41	**99.96**	74.04	67.86	59.65	**85.31**

Table 4: AUC Score of LOF, KNN, DN2 and LUNAR for different values of k and the Avg. over all k. Best performance for each is highlighted in bold.

	Negative sampling scheme		
Dataset	SP	U	Mixed
HRSS	**93.32**	66.34	92.17
MI-F	84.17	57.76	**84.37**
MI-V	96.64	67.99	**96.73**
OPTDIGITS	93.81	**99.86**	99.76
PENDIGITS	99.78	**99.82**	99.81
SATELLITE	**85.37**	85.12	85.35
SHUTTLE	99.96	99.54	**99.97**
THYROID	**85.99**	45.42	85.44

Table 5: AUC scores for different negative sample types.

except for the OPTDIGITS dataset, in which SP samples alone gives poor performance for small values of k. Overall, mixing both types gives the best performance in most cases.

Further ablation studies relating to the neural network size and depth can be found in the supplementary material. Overall, we find that deeper and wider networks for message aggregation give the best performance.

Conclusion

We have studied local outlier methods, some of the most well-established and popular anomaly detection methods in practice, which use the distance of data samples to their nearest neighbours to detect anomalies. We provided a unifying framework which shows that many local outlier methods seen as particular cases of the message passing scheme used in graph neural networks.

We then proposed LUNAR, which is based on this shared framework but is also able to learn and adapt to different sets of data by using a graph neural network. We show that our method significantly outperforms the baselines, including other deep learning-based methods, on a wide variety of datasets. Our method also maintains its strong performance for different neighbourhood sizes much better than other local outlier methods, as it is unique in its ability to learn from all incoming information from the neighbours.

Acknowledgements

References

Abati, D.; Porrello, A.; Calderara, S.; and Cucchiara, R. 2019. Latent space autoregression for novelty detection. In *ICCV*, 481–490.

Amarbayasgalan, T.; Jargalsaikhan, B.; and Ryu, K. H. 2018. Unsupervised novelty detection using deep autoencoders with density based clustering. *Applied Sciences*, 8(9): 1468.

An, J. 2015. Variational Autoencoder based Anomaly Detection using Reconstruction Probability. In *SNU Data Mining Center 2015-2 Special Lecture on IE*.

Angiulli, F.; and Pizzuti, C. 2002. Fast outlier detection in high dimensional spaces. In *European conference on principles of data mining and knowledge discovery*, 15–27. Springer.

Bergman, L.; Cohen, N.; and Hoshen, Y. 2020. Deep nearest neighbor anomaly detection. *arXiv preprint arXiv:2002.10445*.

Beyer, K.; Goldstein, J.; Ramakrishnan, R.; and Shaft, U. 1999. When is "nearest neighbor" meaningful? In *International conference on database theory*, 217–235. Springer.

Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; and Sander, J. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 93–104.

Cai, L.; Chen, Z.; Luo, C.; Gui, J.; Ni, J.; Li, D.; and Chen, H. 2020. Structural temporal graph neural networks for anomaly detection in dynamic graphs. *arXiv preprint arXiv:2005.07427*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 1597–1607. PMLR.

Cui, Z.; Henrickson, K.; Ke, R.; and Wang, Y. 2019. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 21(11): 4883–4894.

Deng, A.; and Hooi, B. 2021. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4027–4035.

Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X.; et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, 226–231.

Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; and Yin, D. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference*, 417–426.

Feng, W.; and Han, C. 2015. A Novel Approach for Trajectory Feature Representation and Anomalous Trajectory Detection. *ISIF*, 1093–1099.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, 1263–1272. PMLR.

Goodge, A.; Hooi, B.; Ng, S. K.; and Ng, W. S. 2020. Robustness of Autoencoders for Anomaly Detection Under Adversarial Impact. *IJCAI*.

Jin, W.; Tung, A. K. H.; Han, J.; and Wang, W. 2006. Ranking outliers using symmetric neighborhood relationship. In *PAKDD*, 577–593. Springer.

Kim, K. H.; Shim, S.; Lim, Y.; Jeon, J.; Choi, J.; Kim, B.; and Yoon, A. S. 2019. RaPP: Novelty Detection with Reconstruction along Projection Pathway. In *ICLR*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kriegel, H.-P.; Kröger, P.; Schubert, E.; and Zimek, A. 2009a. LoOP: local outlier probabilities. In *Proceedings of the 18th ACM conference on Information and knowledge management*, 1649–1652.

Kriegel, H.-P.; Kröger, P.; Schubert, E.; and Zimek, A. 2009b. Outlier detection in axis-parallel subspaces of high dimensional data. In *Pacific-asia conference on knowledge discovery and data mining*, 831–838. Springer.

Liu, F. T.; Ting, K. M.; and Zhou, Z.-H. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*, 413–422. IEEE.

Liu, Y.; Li, Z.; Zhou, C.; Jiang, Y.; Sun, J.; Wang, M.; and He, X. 2019. Generative adversarial active learning for unsupervised outlier detection. *IEEE Transactions on Knowledge and Data Engineering*, 32(8): 1517–1528.

Malhotra, P.; Vig, L.; Shroff, G.; and Agarwal, P. 2015. Long short term memory networks for anomaly detection in time series. In *ESANN*, 89–94. Presses universitaires de Louvain. ISBN 2875870157.

Nicolau, M.; McDermott, J.; et al. 2016. A hybrid autoencoder and density estimation model for anomaly detection. In *International Conference on Parallel Problem Solving from Nature*, 717–726. Springer.

Papadimitriou, S.; Kitagawa, H.; Gibbons, P. B.; and Faloutsos, C. 2003. Loci: Fast outlier detection using the local correlation integral. In *Proceedings 19th international conference on data engineering (Cat. No. 03CH37405)*, 315–326. IEEE.

Ruff, L.; Vandermeulen, R.; Goernitz, N.; Deecke, L.; Siddiqui, S. A.; Binder, A.; Müller, E.; and Kloft, M. 2018. Deep one-class classification. In *ICML*, 4393–4402.

Sakurada, M.; and Yairi, T. 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *MLSDA*, 4. ACM. ISBN 1450331599.

Schlegl, T.; Seeböck, P.; Waldstein, S. M.; Schmidt-Erfurth, U.; and Langs, G. 2017. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *IPMI*, 146–157. Springer.

Schölkopf, B.; Platt, J. C.; Shawe-Taylor, J.; Smola, A. J.; and Williamson, R. C. 2001. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7): 1443–1471.

Sipple, J. 2020. Interpretable, multidimensional, multimodal anomaly detection with negative sampling for detection of

device failure. In *International Conference on Machine Learning*, 9016–9025. PMLR.

Tang, J.; Chen, Z.; Fu, A. W.-C.; and Cheung, D. W. 2002. Enhancing effectiveness of outlier detections for low density patterns. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 535–548. Springer.

Vu, H. S.; Ueta, D.; Hashimoto, K.; Maeno, K.; Pranata, S.; and Shen, S. M. 2019. Anomaly Detection with Adversarial Dual Autoencoders. *arXiv preprint arXiv:1902.06924*.

Zenati, H.; Foo, C. S.; Lecouat, B.; Manek, G.; and Chandrasekhar, V. R. 2018. Efficient gan-based anomaly detection. *arXiv preprint arXiv:1802.06222*.

Zhao, Y.; Deng, B.; Shen, C.; Liu, Y.; Lu, H.; and Hua, X.-S. 2017. Spatio-temporal autoencoder for video anomaly detection. In *ACM Multimedia*, 1933–1941.

Zhao, Y.; Nasrullah, Z.; and Li, Z. 2019. PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research*, 20(96): 1–7.

Zheng, L.; Li, Z.; Li, J.; Li, Z.; and Gao, J. 2019. AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN. In *IJCAI*, 4419–4425.

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