Centralised vs decentralised anomaly detection: when local and imbalanced data are beneficial

Mirko Nardi MIRKO.NARDI@SNS.IT
Scuola Normale Superiore
Pisa, Italy

Lorenzo Valerio LORENZO.VALERIO@IIT.CNR.IT
Andrea Passarella ANDREA.PASSARELLA@IIT.CNR.IT
IIT-CNR
Pisa, Italy

Editors: Nuno Moniz, Paula Branco, Luís Torgo, Nathalie Japkowicz, Michał Woźniak and Shuo Wang.

Abstract

In this paper, we address the problem of anomaly detection in decentralised settings. We took inspiration from the current edge computing trend, pushing towards the development of decentralised ML algorithms, i.e., the devices that collected or generated data are in charge of collaborating to train the ML models without sharing raw data . The challenges connected to this scenario are (i) data distributions of local datasets might be different, (ii) data is very often unlabelled, and (iii) devices have limited computational resources. We address them by proposing an unsupervised ensemble method for decentralised anomaly detection where the base learners are lightweight autoencoders. We aim to investigate whether an ensemble of lightweight models trained in isolation on non-IID and unlabelled local data can compete with heavier models trained in centralised settings. In a task of multi-category anomaly detection, our results show that our method exploits the data imbalance successfully to make accurate predictions.

Keywords: centralised vs decentralised, unsupervised anomaly detection, data imbalance, autoencoders ensemble

1. Introduction

Anomaly detection (AD) (Chandola et al., 2009) addresses the problem of identifying instances of rare events (i.e., anomalies) that are inconsistent for the majority of data considered as *normal*. The concept of *anomaly* might take several different meanings, depending on the reference domain, e.g., Health, Manufacturing, Finance, Networking, to mention a few.

Regardless of the specific domain, there is a growing interest in how and where data is processed to train machine learning (ML) models (Verbraeken et al., 2020). Specifically, due to the explosion of the number of devices at the edge of the internet generating and collecting data[1], the knowledge extraction process is shifting from centralisation to decentralisation. The reasons for such a paradigm shift are both technological and privacy related. From the technological standpoint, it is foreseen that the amount of data generated at the edge will

1. Cisco Annual Internet Report (2018–2023) White Paper

be so massive and often ephemeral that full centralisation might result simply infeasible nor ecological, i.e., it is shown that the carbon footprint of cloud datacentres is becoming an issue and international regulators are pushing toward the development of solutions alternative to the cloud. Moreover, data centralisation raises privacy and ownership issues. More and more attention is given to how the current cloud-based AI systems endanger privacy and data ownership. AI-based service providers require users to upload their data to remote cloud facilities for processing. In response to these challenges, lately appeared several ML approaches suitable for processing data directly where they are generated or collected. This is the case, for example, of the Federated Learning framework (McMahan et al., 2016) and all its derivations (Kairouz, 2021).

When processing data in a completely decentralised fashion, it becomes of paramount importance considering several aspects that are not so relevant in centralised settings. First, local data collected by devices, although belonging to the same domain, might be represented differently at each device, i.e., non IID data distributions. Second, the devices at the edge are not as powerful as their counterpart in the cloud. Therefore, they might not be able to train the same *heavy* models as in the cloud. Third, data labelling at the edge might be an issue, especially when they are collected by automated machines without a human supervisor.

In this paper, we address all these challenges within the framework of anomaly detection. Precisely we want to address the following research questions:

- How to accomplish an unsupervised anomaly detection task in completely decentralised settings, without sharing raw data and relying only on lightweight models with performance comparable to a centralised solution?
- To what extent the data imbalance at the edge devices affects or favours the decentralised anomaly detection task?

We answer these questions considering an instance of multi-class anomaly detection[2] where each device trains a local and lightweight autoencoder (AE). The AEs are then used to build an ensemble that, through a simple heuristic, can discern anomalous data from normal ones, even when the different elements of the ensemble are trained on heterogeneously distributed data. In the paper, we show that our decentralised ensemble is accurate and robust to outliers as long as the local data imbalance enables each device to specialise on a restricted set of categories. In these cases, the advantages of the approach are substantial: the training process can be easily parallelised, the amount of information exchanged is significantly less than the one that would be exchanged by centralising the data.

The remainder of the paper is organised as follows. In Section 2 an overview of the problem and the related works are discussed. In Section 3 and Section 4 we list the preliminaries and we describe our method in detail. In Section 5 we discuss the results of the experiments and in Section 6 we draw the conclusions

2. Related Works

Centralised Anomaly detection Anomaly detection, also referred as *outlier detection* in the literature, is an intrinsically unsupervised problem. Given its wide application, countless

2. Normal data might belong to multiple categories

techniques for this problem have been introduced. Most of the solutions are meant for centralised scenarios and they can be of different kinds, i.e., distance-based, density-based, or subspace based (Aggarwal, 2017). In recent works the focus has shifted to neural network based approaches (Chalapathy and Chawla, 2019), in particular, autoencoder has revealed to be a simple yet effective tool for this task (Xia et al., 2015) since it is able to employ nonlinear dimensionality reductions (Sakurada and Yairi, 2014). On top of these approaches, several ensemble methods have been proposed (Aggarwal and Sathe, 2017), allowing remarkable improvements in many aspects depending on the specific case. In addition, recent works has been introduced involving autoencoder ensembles for unsupervised tasks (Chen et al., 2017; Sarvari et al., 2019).

Decentralised Anomaly detection Unsupervised tasks in distributed contexts is a far less studied topic compared to its supervised counterpart. The gap between them is even larger if modern supervised NN-based collaborative paradigms like federated learning are taken into account (Jin et al., 2020). Relevant studies of unsupervised distributed learning have been made within anomaly detection in wireless sensor networks, targeting challenges similar to those discussed in Section 1. However, these works are generally based on more traditional approaches like clustering methods or predefined statistical assumptions each with their own limitations in terms of high computational complexity, communication overhead or lack of collaboration between devices, as noted by Rajasegarar et al. (2006); Luo and Nagarajany (2018). The first effort of introducing a collaborative system of autoencoders for distributed anomaly detection is given by Luo and Nagarajany (2018). However, the lightweight autoencoders adopted in the solution are trained in cloud (i.e., centralised training) on data collected by small devices. Locally, the autoencoders are used for inference only.

Precisely, we address the problem of performing anomaly detection based on an ensemble of autoencoders trained without complete knowledge of the whole dataset. Moreover, as an additional challenges we impose that the autoencoders must be lightweight. Our interest is to investigate what are the limits of such a process in both IID and non-IID settings. To the best of the authors' knowledge, this problem has not yet been explored in literature.

3. Preliminaries

In anomaly detection the goal is to train a model that recognises anomalous data out of normal ones. The central assumption in anomaly detection is that normal data outnumbers anomalies that are considered by definition *rare events*. An anomaly detection algorithm (Aggarwal, 2017) generally proceeds as follow: (i) it builds a model of the normal patterns in the data; (ii) it computes an anomaly/outlier score of a given test sample based on the deviation from the normal patterns. (iii) The anomaly scores provided by an AD algorithm can be converted into binary labels, indicating whether a data point is an anomaly or not. This is typically achieved by imposing thresholds on outlier scores; a threshold can be chosen based on the statistical properties of the scores.

In the paper, we use autoencoders as base models. We found autoencoder particularly interesting for their flexibility regardless of the domain and the dimensionality of the data. Moreover, although not covered in this work, they are suitable for incremental training, differently from the vast majority of other approaches adopted in AD. An autoencoder is a multi-layer neural network specifically designed to learn a compressed representation of unlabelled data by performing a nonlinear dimensionality reduction. It has a symmetric structure in which an encoder network maps the d-dimensional input into a latent k-dimensional code with $k < d$, and a decoder network recovers the data from the code. The goal is to reconstruct the input as closely as possible to itself. Formally, given a training set $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \in \mathbb{R}^d$ such that $\boldsymbol{x}_i$ is a d-dimensional input sample (i.e, a feature vector), the autoencoder is a composite function $f(\boldsymbol{x}_i) := h_{\theta'}(g_\theta(\boldsymbol{x}_i))$ where $g : \mathbb{R}^d \to \mathbb{R}^k$ and $h : \mathbb{R}^k \to \mathbb{R}^d$ are the encoder and the decoder functions parameterized by $\boldsymbol{\theta}, \boldsymbol{\theta'}$, respectively. For the sake of readability, we denote the set of parameters $\boldsymbol{\theta}, \boldsymbol{\theta'}$ as $\boldsymbol{\Theta}$. The reconstruction of the input is then indicated as $\boldsymbol{x}'_i = f_{\boldsymbol{\Theta}}(\boldsymbol{x}_i)$.

The square loss $e_i = \|f_\theta(\boldsymbol{x}_i) - \boldsymbol{x}_i\|^2$ is usually taken as reconstruction error of $\boldsymbol{x}_i$, therefore, the autoencoder is trained[3] by finding the weights $\boldsymbol{\Theta}$ that minimise the average reconstruction error: $\mathcal{L}(f_\Theta) = 1/n \sum_{i=1}^{n} \|f_{\boldsymbol{\Theta}}(\boldsymbol{x}_i) - \boldsymbol{x}_i\|^2$.

When a sample is given as input to a trained autoencoder, both the reduced dimensional representation (i.e., the output of the middle layer) or the reconstruction error can be exploited to understand whether it is an outlier or not.

The rationale is that during training, the autoencoder captures statistical regularities of the training set. Thus, when an anomalous sample occurs, the corresponding reconstruction error will be higher than normal.

4. Proposed Methodology

Our distributed system is composed by K devices. Each device is equipped with a copy of the same autoencoder which is trained on a partition D_i of the dataset D, such that $D = \bigcup_{i=1}^{K} D_i$ where $D_i \neq D_j$ for $i \neq j$.

The idea is that if the data partitioning allows the devices to specialise individually on a specific normal class, then, once all the autoencoders are collected, a data point can be scored by each autoencoder by computing the reconstruction error. The autoencoder that provides the minimum reconstruction error is the one that most likely recognised it as normal. A data point is recognised as an anomaly in our scheme when no device in the ensemble recognises it.

Formally, providing a test data point $\boldsymbol{x}$ to the ensemble of K autoencoders $\{AE_i\}_{i=1}^{K}$, we obtain a set $\boldsymbol{s} = \{s_1, \dots, s_K\}$ of K reconstruction errors. The elements of the vector $\boldsymbol{s}$ represent the *anomaly scores* assigned by each AE to the point $\boldsymbol{x}$. In our solution the final aggregate score s^* corresponds to the minimum reconstruction error over the different ensemble components, as in Eq. (1).

$$s^* = \min\{\|AE_i(\boldsymbol{x}) - \boldsymbol{x}\|^2\}_{i=1,\dots,K} \tag{1}$$

Finally, the AE corresponding to the minimum score, here denoted as AE^*, is the one in charge of providing also the binary decision on the test point $\boldsymbol{x}$ as in Eq. (2).

$$b(s^*, \tau_{AE^*}) = \begin{cases} 1 & s^* > \tau_{AE^*} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

3. Training is performed through back-propagation of the error.

where τ_{AE^*} is a fixed threshold locally computed by AE^*. Here one stands for anomaly and zero for normal. The threshold embeds a prior assumption about the expected percentage of outliers ρ that might be present in the whole data (not in the local partitions). Each autoencoder exploits ρ to calculate its threshold as the $(100 - \rho)$th percentile of its training scores, i.e., the reconstruction errors computed on the local training set. We motivate this choice considering that in a completely decentralised setting where sharing is not allowed, each device can only leverage the local knowledge build on its local data and some common prior knowledge. This ensemble-based anomaly detection method is reported in Algorithm 1.

Algorithm 1: DAE - Decentralised Anomaly Detection

```
Input: $x_{test}, K, \{AE_i\}_{i=1}^K, \{\tau_i\}_{i=1}^K$
$\boldsymbol{s} \leftarrow [\,]$
for $i \leftarrow 1$ to $K$ do
    $s_i \leftarrow \|AE_i(x_{test}) - x_{test}\|^2$ ;          ▷ Get reconstruction error from each AE
end
$i^* \leftarrow \arg\min_i(\boldsymbol{s})$ ;          ▷ Get the index of the AE with minimum score
return $b(s_{i^*}, \tau_{i^*})$ ;          ▷ return the binary response
```

5. Numerical results

In this section, we evaluate the performance of our decentralised ensemble-based anomaly detection system. We recall that our research questions address two main aspects. On the one hand, we investigate if an ensemble of lightweight autoencoders trained in isolation on non-IID data can achieve performance comparable to a centralised and potentially heavier model trained by centralising all the data.

On the other hand, we investigate to what extent the level of data heterogeneity affects the performance of our decentralised method. For the sake of reproducibility, our code is based on well-accessed and standard frameworks: Tensorflow[4], Scikit-Learn[5] and PyOD[6]. The code is available at `https://github.com/mirqr/MultiAEDist`.

5.1. Dataset preparation and performance metrics

In our experiments we use `MNIST` (LeCun et al., 2010) and `fashion-MNIST` (Xiao et al., 2017), two widely used labelled data sets that are commonly adapted to the anomaly detection scenario as well. They share the same structure, consisting of a training set of 60000 examples and a test set of 10000 examples. Each example is a 28×28 grey-scale image associated with a label from 10 classes. The former contains handwritten digits, the latter contains images of clothing and it represents a considerably more complex learning task[7]. We use the original train/test splits and exploit the data labels to perform a comprehensive set of experiments in

4. https://www.tensorflow.org
5. https://scikit-learn.org/stable/
6. https://pyod.readthedocs.io/en/latest/
7. `MNIST` vs. `fashion-MNIST`: `http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/`

which we compare several multi-class distributed scenarios to the corresponding centralised ones.

For each experiment, from the set of classes C we define a subset inlier classes C_I of size S_I such that $S_I \geq 2$. All the training samples belonging to C_I are labelled as normal, while the samples of the remaining classes $C_{OUT} = C \setminus C_I$ are labelled as anomalies. Remember that in our settings, the anomaly detection problem is unsupervised. Therefore, we use the labels for evaluation purposes only.

To simulate data imbalance among devices' local datasets, we adopt the following procedure. Given the number of inlier classes S_I, we partition the whole dataset D into S_I disjoint partitions D_i, such that $D = \bigcup_{i=1}^{S_I} D_i$ where $D_i \neq D_j \forall i \neq j$. Moreover, we impose that the $\mathcal{D}_i \neq \mathcal{D}_j \forall i, j = 1, \ldots, S_I$, where $\mathcal{D}_i$ is the distribution of the i-th data partition D_i. For the rest of the paper we set $K = S_I$. We consider a range of data partitioning configurations. At one extreme, we assume that each D_i contains only samples from a single normal class (from now on called *ideal partitioning*). At the other extreme, at each local datasets D_i, the normal classes are equally represented. This is a limit case that we include for completeness to cover the situation where local datasets are IID. We term it *uniform partitioning*. Finally, for the middle cases, we allocate the samples to the D_i partitions according to a discrete Zipf$(n = S_I, a)$ distribution under different values of the parameter a (i.e., the exponent of the distribution). As a result, each device takes most of its samples from one normal class and few samples from all the remaining normal classes. To preserve heterogeneity between the local datasets D_i, we ensure that the Zipf ranking of the normal classes is circular along with the devices. Figure 1(a) and Figure 1(b) exemplify the data distributions at each device in the cases of *ideal partitioning* and *Zipf-like partitioning*, respectively. Note that we use the labels information only to partition the data: once placed, all the samples are deprived of labels.

Beyond the three macro-cases of data partitioning, to test the robustness of our decentralised anomaly detection solution to outliers, we design two separate sets of experiments. In the first set, we considered local datasets free from any outliers. This is an ideal case we use to set the baseline. In the second set, we assume that each local dataset D_i contains a fixed percentage of outliers (i.e., 10% as it is commonly assumed in the anomaly detection literature). Here the outliers are uniformly sampled from the patterns of C_{OUT}. In the distributed cases, they are uniformly spread among the devices. Hence the training inliers/outliers ratio is preserved in both centralised and distributed settings.

The performance evaluation is done comparing the accuracy of the decentralised anomaly detection in all the data partition configurations with the one obtained by a centralised anomaly detection solution with complete access to the whole training set D. In real-world settings, this represents a scenario where data collected by the decentralised devices are collected on a central server for being processed. The data distribution of D is exemplified in Figure 1(c).

We measure the accuracy of both the decentralised and centralised anomaly detection algorithms on a common test set by using two standard metrics: the F_1 as accuracy score, considering the outlier label as the positive label, and the Receiver Operating Characteristic (ROC) curve to generate the full trade-off between the true positive rate and the false positive rate, which we summarise in a single value by computing the area under the ROC curve (AUC).

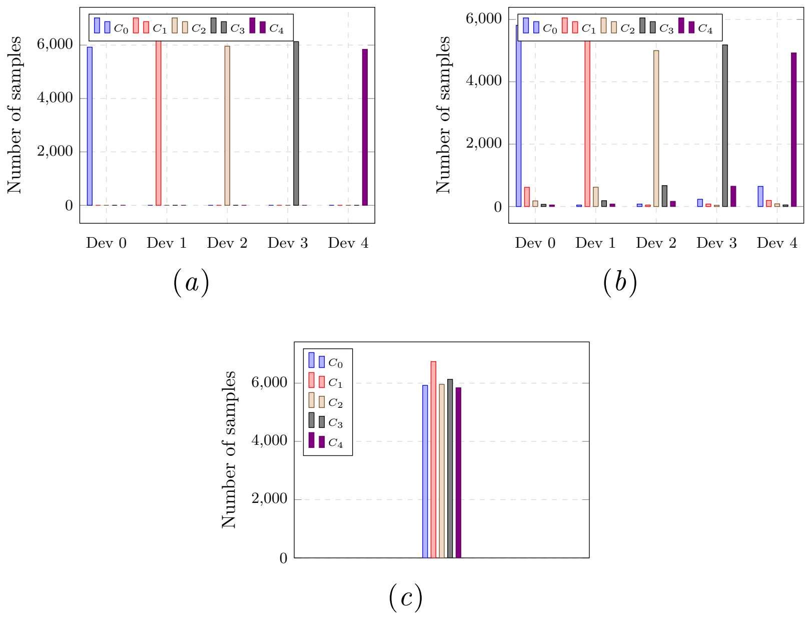

Figure 1: Examples of data partitioning using MNIST with $C_I = \{0, 1, 2, 3, 4\}$. (a) Ideal distributed partitioning, (b) Zipf($n = 5, a = 3$) distributed partitioning and (c) uniform centralised settings.

5.2. Experimental settings

Centralised anomaly detection benchmarks. The centralised benchmarks considered in this paper are three state-of-the-art algorithms commonly used in anomaly detection, namely, One-class Support Vector Machines (OC-SVM), Local Outlier Factor (LOF) and a Deep Autoencoder (FAE).

OC-SVM (Schölkopf et al., 1999) is a special case of support vector machine, particularly effective in semi-supervised tasks, i.e. assumes only positively labelled instances in the training data. It maps input data into a high dimensional feature space and iteratively finds the maximal margin hyperplane that best separates the origin's training data. It requires the choice of a kernel and a scalar parameter $\nu \in (0, 1]$ to define a frontier. It can be used with contaminated data set but requires fine-tuning of ν to prevent over-fitting, which can be challenging without any assumptions on the distribution of the outliers. In our experiments, we use the RBF kernel and set $\nu = 0.1$, an upper bound on the fraction of training errors and a lower bound on the fraction of support vectors.

LOF (Breunig et al., 2000) is a conventional distance-based algorithm originally meant to be applied for unsupervised outlier detection tasks. It computes the local density deviation of a given data point and detects the ones that have a substantially lower value than their neighbours. The local density is obtained from the k-nearest neighbours, i.e. k is the main parameter of the estimator. We set the number of nearest neighbours $k = 20$, after testing other values from 10 to 20 as suggested in the original paper. Since LOF is originally meant for outlier detection tasks, it also needs a *contamination* parameter to be fixed, representing the proportion of outliers in the data set used to define the threshold on the anomaly scores of the samples. We set it to 0.1, a commonly used value when the training set is uncontaminated.

FAE is a deep fully connected autoencoder based method. This recent approach, along with its numerous extensions, has proved to stand up to classical algorithms by exploiting the nonlinear representation of the data. For the definition of the number of the hidden layer, we followed a strategy similar to (Chen et al., 2017): we set the number of neurons of the middle layer to 16, a reasonable value to avoid an excessive information bottleneck, and we doubled it in each added layer until we reach the closest number of neurons of the input/output layer (e.g., 512 for MNIST, since it is 784 dimensions). We tested all structures obtained, and we observed that after seven hidden layers (i.e., 128 neurons for the outermost hidden layers), the accuracy does not significantly improve due to the tendency of the neural network to overfit. Hence, for the centralised part we selected three architectures (denoted as FAE3, FAE5, FAE7) with the following hidden layers' structure: [32-16-32], [64-FAE3-64], [128-FAE5-128], respectively.

Decentralised autoencoders. The autoencoders forming the decentralised ensemble have the same architecture as FAE3. FAE3 is a lightweight autoencoder with only $50k$ parameters that resource-limited devices might relatively easily train. We included it also among the centralised experiments to have a comparison on the same model. The remaining parameters and settings are the same among all autoencoders: the activation functions of input and output layers are sigmoid, while for hidden layers, we set ReLU. This is a classic combination that turned out to be resilient to the vanishing gradient problem. We used Adam (Kingma and Ba, 2015) as the optimizer with learning rate $\eta = 10^{-3}$ tuned over the set $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$.

Table 1: Inlier classes used in the experiments.

	Normal Classes
C_{I1}	$\{0, 1, 4, 6, 9\}$
C_{I2}	$\{1, 2, 3, 4, 8\}$
C_{I3}	$\{0, 1, 6, 7, 9\}$
C_{I4}	$\{0, 3, 5, 6, 8\}$
C_{I5}	$\{3, 4, 6, 7, 8\}$
C_{I6}	$\{0, 2, 3, 4, 5\}$
C_{I7}	$\{0, 1, 2, 3, 4\}$
C_{I8}	$\{5, 6, 7, 8, 9\}$
C_{I9}	$\{0, 2, 4, 6, 8\}$
C_{I10}	$\{1, 3, 5, 7, 9\}$

5.3. Results

We run ten experiments for both the MNIST (MN) and fashion-MNIST (F-MN) data sets, each time randomly selecting a set of five inlier classes to form C_I as shown in Table 1. From now on, we refer to our ensemble of lightweight autoencoders, each one trained in isolation on one of the D_i local datasets as *Decentralised Autoencoder Ensemble (DAE)* and the other anomaly detection solutions trained on the whole datasets as *centralised solutions*.

In Table 2 we compare how DAE performs against all the centralised solutions, under ideal-partitioning where data do not include outliers (DAE Ideal). For the MNIST data set,

Table 2: AUC and F1 results on MNIST (MN) and fashion-MNIST (F-MN) with no contamination. Comparison of centralised methods against a perfectly distributed setting (the rightmost columns). The best overall scores are highlighted in boldface. The best centralised scores are underlined.

Dataset	AUC						$F1$					
	OCSVM	LOF	FAE3	FAE5	FAE7	DAE Ideal.	OCSVM	LOF	FAE3	FAE5	FAE7	DAE Ideal.
MN C_{I1}	0.818	**0.976**	0.957	0.966	0.974	0.97	0.656	**0.935**	0.891	0.903	0.913	0.904
MN C_{I2}	0.722	**0.926**	0.819	0.845	0.89	0.915	0.498	**0.836**	0.617	0.680	0.774	0.832
MN C_{I3}	0.756	0.971	0.963	0.972	0.98	**0.985**	0.586	0.927	0.899	0.919	0.922	**0.933**
MN C_{I4}	0.657	**0.905**	0.858	0.866	0.868	0.894	0.336	0.775	0.686	0.686	**0.789**	0.752
MN C_{I5}	0.629	**0.899**	0.725	0.76	0.791	0.859	0.382	**0.747**	0.56	0.608	0.683	0.73
MN C_{I6}	0.553	**0.872**	0.680	0.705	0.75	0.833	0.199	**0.748**	0.401	0.489	0.592	0.714
MN C_{I7}	0.624	**0.938**	0.848	0.884	0.908	0.92	0.317	**0.867**	0.648	0.743	0.798	0.84
MN C_{I8}	0.695	**0.918**	0.831	0.853	0.869	0.88	0.463	0.774	0.658	0.753	**0.792**	0.766
MN C_{I9}	0.666	**0.867**	0.715	0.733	0.806	0.841	0.319	**0.727**	0.481	0.513	0.633	0.688
MN C_{I10}	0.795	0.953	0.962	0.963	0.973	**0.976**	0.648	0.901	0.898	0.904	**0.916**	0.91
avg.	0.691	**0.922**	0.836	0.855	0.881	0.907	0.44	**0.824**	0.674	0.720	0.781	0.807
F-MN C_{I1}	0.742	0.836	0.824	0.832	0.836	**0.841**	0.625	0.603	0.658	0.683	**0.691**	0.688
F-MN C_{I2}	0.752	0.845	0.831	0.832	0.839	**0.854**	0.703	0.675	0.682	0.698	**0.709**	0.617
F-MN C_{I3}	0.646	**0.838**	0.816	0.825	0.826	0.833	0.499	0.604	0.567	0.599	**0.619**	0.597
F-MN C_{I4}	0.588	**0.692**	0.637	0.654	0.656	0.677	0.303	0.339	0.265	0.303	0.306	**0.362**
F-MN C_{I5}	0.687	**0.774**	0.720	0.717	0.723	0.755	0.538	0.575	0.456	0.477	0.505	**0.622**
F-MN C_{I6}	0.677	0.72	0.757	0.772	**0.772**	0.768	0.477	0.458	0.543	0.572	**0.59**	0.537
F-MN C_{I7}	0.875	0.902	0.913	0.91	0.914	**0.917**	0.849	0.826	**0.851**	0.847	0.848	0.845
F-MN C_{I8}	0.64	**0.766**	0.682	0.655	0.661	0.682	0.507	**0.535**	0.436	0.406	0.406	0.401
F-MN C_{I9}	**0.904**	0.874	0.897	0.881	0.876	0.882	**0.809**	0.657	0.776	0.736	0.735	0.678
F-MN C_{I10}	0.66	0.834	0.891	0.891	0.877	**0.909**	0.61	0.614	0.774	0.778	0.753	**0.818**
avg.	0.717	0.808	0.797	0.797	0.798	**0.812**	0.592	0.589	0.601	0.610	0.615	**0.616**

DAE has the second-best results after the LOF algorithm, with an average AUC score of 0.907 against 0.922 respectively. On MNIST, LOF appears to work particularity well, while OC-SVM has difficulty finding a normality pattern in an intrinsic multi-class training set. At the same time, by using a more complex data set like fashion-MNIST, DAE can produce the highest AUC and F1 scores. The autoencoders used in the centralised setting (FAE) are ranked in the middle, and we note that at least 7-layers (FAE7) are needed to get a score reasonably close to the distributed solution.

Similar considerations can be made by introducing a 10% of contamination to the training set, as shown by the results in Table 3. Here DAE overtakes all the centralised models in both MNIST and fashion-MNIST data sets, proving a good tolerance to outliers. LOF struggles to find clear density variations; hence it performs poorly, especially on fashion-MNIST. The second-bests are FAE7 for MNIST and FAE3 for fashion-MNIST, with a 13% and a 6% decrease of the AUC score. Note that we kept the decision threshold to 0.1 in this test case, as in the uncontaminated experiments. Considering the generally low values for $F1$, a threshold of 0.1 is probably insufficient in a multiple categories anomaly detection task with 10% of contamination rate. Nevertheless, the performance ranking of the algorithms remains consistent.

In the second block of our experiments, we compare several distributed settings in order to analyze the sensitivity of DAE when the samples of the inliers classes are not perfectly arranged into different partitions. To simulate such a scenario, we place the samples following a Zipf distribution using $a = \{3, 2.5, 2\}$, ensuring that the ranking of the inlier classes is circular among the devices (Figure 1(b)). In this way, we make sure each device is still exposed to a slightly more heterogeneous data distribution than in the previous ideal case.

Table 3: AUC and F1 results on MNIST (MN) and fashion-MNIST (F-MN) with 10% of outlier contamination. Comparison of centralised methods against DAE trained in a perfectly distributed setting (the rightmost columns). The best overall scores are highlighted in boldface. The best centralised scores are underlined.

Dataset	AUC						$F1$					
	OCSVM	LOF	FAE3	FAE5	FAE7	DAE Ideal	OCSVM	LOF	FAE3	FAE5	FAE7	DAE Ideal
MN C_{I1}	0.7	0.676	0.870	0.861	0.87	**0.942**	0.422	0.430	0.635	0.647	0.701	**0.712**
MN C_{I2}	0.643	0.557	0.673	0.678	0.684	**0.86**	0.351	0.217	0.285	0.307	0.357	**0.489**
MN C_{I3}	0.65	0.705	0.900	0.914	0.914	**0.957**	0.379	0.425	0.626	0.689	**0.742**	0.719
MN C_{I4}	0.578	0.542	0.597	0.621	0.585	**0.754**	0.233	0.322	0.338	0.424	0.402	**0.474**
MN C_{I5}	0.593	0.591	0.578	0.600	0.615	**0.724**	0.302	0.317	0.282	0.341	0.397	**0.492**
MN C_{I6}	0.519	0.611	0.615	0.642	0.677	**0.747**	0.158	0.337	0.407	0.461	0.564	**0.566**
MN C_{I7}	0.568	0.648	0.826	0.872	0.902	**0.946**	0.226	0.326	0.540	0.640	**0.725**	0.656
MN C_{I8}	0.646	0.603	0.660	0.672	0.643	**0.729**	0.352	0.341	0.398	0.446	0.453	**0.539**
MN C_{I9}	0.585	0.577	0.521	0.515	0.532	**0.681**	0.191	0.289	0.229	0.255	0.311	**0.376**
MN C_{I10}	0.707	0.656	0.866	0.867	0.852	**0.932**	0.473	0.350	0.566	0.600	0.62	**0.678**
avg.	0.619	0.617	0.711	0.724	0.727	**0.827**	0.309	0.336	0.431	0.481	0.527	**0.57**
F-MN C_{I1}	0.673	0.584	0.732	0.726	0.723	**0.789**	0.405	0.238	0.413	0.420	0.42	**0.466**
F-MN C_{I2}	0.644	0.523	0.697	0.695	0.691	**0.779**	0.374	0.211	0.377	0.375	0.393	**0.398**
F-MN C_{I3}	0.6	0.637	0.766	0.767	0.752	**0.811**	0.383	0.293	0.437	0.446	0.445	**0.463**
F-MN C_{I4}	0.55	0.439	0.510	0.491	0.489	**0.593**	**0.212**	0.106	0.153	0.158	0.176	0.145
F-MN C_{I5}	0.633	0.563	0.609	0.599	0.59	**0.653**	0.383	0.250	0.301	0.310	0.314	**0.415**
F-MN C_{I6}	0.621	0.420	0.688	0.678	0.675	**0.698**	0.324	0.121	0.405	0.407	**0.446**	0.3
F-MN C_{I7}	0.751	0.506	0.909	0.910	**0.913**	0.909	0.525	0.176	0.822	0.814	**0.824**	0.766
F-MN C_{I8}	0.517	0.513	0.502	0.444	0.433	**0.534**	**0.265**	0.203	0.124	0.129	0.136	0.164
F-MN C_{I9}	**0.792**	0.465	0.693	0.641	0.626	0.732	**0.417**	0.131	0.337	0.324	0.322	0.363
F-MN C_{I10}	0.486	0.465	0.727	0.689	0.693	**0.792**	0.385	0.180	0.381	0.362	0.383	**0.514**
avg.	0.627	0.511	0.683	0.664	0.658	**0.729**	0.367	0.191	0.375	0.375	0.386	**0.4**

Finally, we added the case in which all samples are uniformly spread among the devices as the worst case (DAE Uniform).

In Table 4 and in Table 5 we show the percentage of gain/loss of DAE compared to the best centralised scores (underlined in Table 2 and Table 3 respectively) for all the Zipf-like cases and uniform data partitioning. Looking at the AUC scores, we observe that in the uncontaminated cases (Table 4) only on fashion-MNIST, the distributed algorithm can compete with the best-centralised ones, recalling that LOF performed very well on MNIST experiments without outliers. The results radically improve in the contaminated cases (Table 5); here, we get an increase in the AUC scores in almost every distributed scenario. It is interesting to note how the improvement decreases as long as the class distribution flattens, i.e. this happens for lower values of a. Of course, there is a drastic degradation of performance in the uniform case since the lightweight autoencoders can no longer specialise in a single inlier class and their small architecture is not expressive enough to generalise all the categories presented in the local dataset. Regarding the F1 metric, in both groups of experiments, we observe that only in the ideally distributed scenario there are improvements (or moderate degradations) in the scores. This can be explained by recalling the binary decision criteria we described in Section 4: every device computes its decision threshold from its local reconstruction errors (on the training set) and a *fixed* contamination parameter. In a non-perfectly distributed situation like the Zipf cases, every device has most of its samples from one inlier class and a small chunk from other inlier classes, which forms an additional portion of local outliers. Since we made no assumptions on it, the fixed contamination value works only in an ideally distributed case. Nevertheless, the high values of the AUC scores

Table 4: Gain/loss comparison (%) between the best centralised scores (underlined in Table 2) and DAE under various distribution settings on MNIST and fashion-MNIST (no contamination).

Training Set	AUC gain/loss (%)					$F1$ gain/loss (%)				
	Ideal	Zipf			Uniform	Ideal	Zipf			Uniform
		$a = 3$	$a = 2.5$	$a = 2$			$a = 3$	$a = 2.5$	$a = 2$	
MN C_{I1}	-0.615	-1.025	-1.025	-1.639	-2.459	-3.316	-7.594	-5.989	-8.663	-7.594
MN C_{I2}	-1.188	-4.32	-3.78	-3.888	-12.419	-0.478	-19.258	-22.727	-28.708	-36.124
MN C_{I3}	**0.51**	-0.408	-0.408	-0.51	-2.245	**0.647**	-3.236	-2.697	-4.099	-8.091
MN C_{I4}	-1.215	**0.552**	-2.652	-2.21	-7.956	-4.689	-12.041	-17.490	-17.871	-29.024
MN C_{I5}	-4.449	-7.564	-9.455	-10.011	-21.246	-2.276	-19.009	-24.498	-31.861	-39.491
MN C_{I6}	-4.472	-8.257	-8.83	-11.468	-21.674	-4.545	-30.214	-43.984	-50.668	-60.027
MN C_{I7}	-1.919	-1.919	-1.812	-5.224	-11.940	-3.114	-24.913	-26.528	-35.409	-43.137
MN C_{I8}	-4.139	-4.793	-5.882	-8.388	-11.111	-3.283	-11.995	-15.530	-21.465	-24.621
MN C_{I9}	-2.999	-9.573	-8.304	-8.42	-23.529	-5.365	-38.377	-48.418	-47.868	-60.385
MN C_{I10}	**0.308**	-0.719	-1.336	-0.719	-1.953	-0.655	-3.930	-5.240	-6.987	-6.004
F-MN C_{I1}	**0.598**	-1.196	**0.239**	**0.239**	-1.555	-0.434	-16.353	-15.051	-15.630	-11.577
F-MN C_{I2}	**1.065**	**0.355**	**0.0**	**0.118**	-0.237	-12.976	-17.772	-14.386	-17.066	-2.257
F-MN C_{I3}	-0.597	-1.79	**0.835**	**0.119**	-1.551	-3.554	-17.447	-16.801	-20.679	-11.793
F-MN C_{I4}	-2.168	-4.913	-5.491	-4.48	-7.803	**6.785**	-15.339	-30.088	-37.463	-35.103
F-MN C_{I5}	-2.455	-4.522	-5.814	-4.134	-7.364	**8.174**	-20.522	-17.391	-21.391	-25.391
F-MN C_{I6}	-0.518	-1.554	-1.036	-1.295	-1.036	-8.983	-21.186	-18.644	-21.017	-14.746
F-MN C_{I7}	**0.328**	**0.219**	**0.0**	-0.109	-0.438	-0.705	-0.118	-0.588	-0.118	-0.588
F-MN C_{I8}	-10.966	-10.574	-8.094	-12.141	-11.488	-25.047	-35.888	-45.421	-38.879	-30.467
F-MN C_{I9}	-2.434	-1.991	-0.885	-2.212	-2.102	-16.193	-18.047	-15.575	-20.272	-13.597
F-MN C_{I10}	**2.02**	**0.337**	**1.01**	**1.796**	-0.337	**5.141**	-6.684	-16.967	-6.170	-2.442

indicate that setting decision thresholds (τ) properly, the distributed algorithm has good potential in the other cases.

Table 5: Gain/loss comparison (%) between the best centralised scores (underlined in Table 3) and DistAE under various distribution settings on MNIST and fashion-MNIST (10% contamination).

Training Set	AUC gain/loss (%)					$F1$ gain/loss (%)				
	Ideal	Zipf			Uniform	Ideal	Zipf			Uniform
		$a = 3$	$a = 2.5$	$a = 2$			$a = 3$	$a = 2.5$	$a = 2$	
MN C_{I1}	**8.276**	**5.977**	**4.943**	**3.563**	**0.115**	**1.569**	-25.678	-25.25	-25.963	-19.829
MN C_{I2}	**25.731**	**18.421**	**14.327**	**7.749**	-2.047	**36.975**	-27.731	-19.048	-30.532	-45.938
MN C_{I3}	**4.705**	**3.72**	**2.735**	**2.298**	-1.422	-3.1	-22.642	-20.889	-22.372	-26.280
MN C_{I4}	**21.417**	**15.62**	**15.298**	**11.272**	**4.348**	**11.792**	-16.745	-26.887	-24.057	-25.708
MN C_{I5}	**17.724**	**12.195**	**11.057**	**8.78**	-4.228	**23.929**	-28.967	-35.768	-35.516	-39.547
MN C_{I6}	**10.34**	**6.204**	**4.727**	**2.068**	-11.669	**0.355**	-36.879	-42.73	-44.504	-54.787
MN C_{I7}	**4.878**	**2.217**	**0.998**	-1.663	-7.539	-9.517	-26.759	-34.897	-35.172	-38.069
MN C_{I8}	**8.482**	**6.548**	**6.994**	**2.232**	-3.274	**18.985**	-15.453	-22.517	-28.477	-39.294
MN C_{I9}	**16.41**	**9.915**	**9.744**	**6.154**	-12.821	**20.9**	-34.084	-30.868	-39.550	-45.659
MN C_{I10}	**7.497**	**5.075**	**3.576**	**2.537**	**0.577**	**9.355**	-10.968	-11.29	-11.452	-22.419
F-MN C_{I1}	**7.787**	**6.284**	**5.601**	**6.011**	**0.82**	**10.952**	-17.857	-15.238	-13.333	-13.810
F-MN C_{I2}	**11.765**	**7.891**	**5.308**	**6.169**	-0.43	**1.272**	-5.344	-1.781	-8.397	-12.723
F-MN C_{I3}	**5.737**	**4.172**	**4.694**	**2.608**	**0.652**	**3.812**	-19.507	-17.265	-21.749	-15.022
F-MN C_{I4}	**7.818**	**4.182**	**0.909**	**1.455**	-5.636	-31.604	-49.057	-45.755	-46.226	-41.981
F-MN C_{I5}	**3.16**	**3.16**	**3.633**	**2.528**	-2.37	**8.355**	-14.883	-14.099	-21.671	-24.021
F-MN C_{I6}	**1.453**	**1.163**	**1.744**	**1.599**	**0.727**	-32.735	-32.511	-29.372	-29.372	-18.610
F-MN C_{I7}	-0.438	-0.329	-0.219	-0.767	-1.752	-7.039	-5.825	-6.432	-8.374	-12.257
F-MN C_{I8}	**3.288**	**2.708**	**3.288**	**0.58**	-3.288	-38.113	-56.604	-58.491	-68.302	-60.755
F-MN C_{I9}	-7.576	-10.354	-11.364	-10.985	-14.52	-12.95	-26.379	-31.894	-26.139	-27.098
F-MN C_{I10}	**8.941**	**6.602**	**4.677**	**4.814**	-0.825	**33.506**	-0.519	**5.974**	-2.857	-15.844

To verify such a claim, we substitute for each device the fixed value of the contamination rate ρ with the real percentage r_1 of local data not belonging to the most represented class, i.e., the class of rank 1 in the Zipf distribution. With this information, the devices could modify their local decision threshold for detecting anomalies by setting $\rho = r_1$ and computing the corresponding percentile. We test this assumption within the same settings of the results presented in Table 5. Looking at Table 6, it is interesting to note that using a locally computed threshold that takes into account the structure of the local dataset, apart from few isolated cases, almost all the F1 scores are positive, meaning that DAE outperforms the best-centralised algorithm. Surprisingly, it appears that using such a simple heuristic, it would be possible to improve the F1 accuracy up to 79.552%. This suggests that it might represent an interesting research direction to investigate in future work.

Table 6: F1 gain/loss between the best centralised scores (underlined in Table 3) and DAE under Zipfs distribution (10% outlier contamination). The r_1 values for $a = \{2, 2.5, 3\}$ are 15.66%, 22.48%, 31.68%, respectively.

Training Set	$F1$		
	DistAE Zipf$_{a\,=\,3}$	DistAE Zipf$_{a\,=\,2.5}$	DistAE Zipf$_{a\,=\,2}$
MN C_{I1}	-1.284	**13.837**	**18.117**
MN C_{I2}	**38.655**	**58.263**	**79.552**
MN C_{I3}	**1.617**	**13.747**	**17.79**
MN C_{I4}	**20.991**	**31.84**	**49.528**
MN C_{I5}	**5.542**	**27.456**	**48.111**
MN C_{I6}	-11.525	**0.532**	**7.979**
MN C_{I7}	-4	**4.138**	**12.414**
MN C_{I8}	**14.57**	**29.801**	**41.943**
MN C_{I9}	**7.074**	**50.161**	**68.167**
MN C_{I10}	**16.613**	**33.71**	**38.871**
F-MN C_{I1}	**11.667**	**34.762**	**52.619**
F-MN C_{I2}	**25.954**	**51.399**	**60.814**
F-MN C_{I3}	**6.951**	**33.408**	**47.534**
F-MN C_{I4}	-8.491	**35.849**	**69.34**
F-MN C_{I5}	**8.355**	**31.332**	**40.47**
F-MN C_{I6}	-8.296	**9.417**	**25.336**
F-MN C_{I7}	**1.82**	**2.063**	**0.485**
F-MN C_{I8}	-27.925	**5.283**	**28.679**
F-MN C_{I9}	-5.036	**11.751**	**45.084**
F-MN C_{I10}	**39.221**	**55.325**	**74.286**

6. Conclusion

In this paper we address the problem of anomaly detection in decentralised settings, meaning that the data is spread over several locations and it can be accessed and processed only locally using resource constrained edge devices. Specifically, we want to address the following open problems: (i) how to perform this unsupervised task without sharing raw data and relying only on lightweight models with performance comparable to a centralised solution; (ii) to what extent the data imbalance at the edge devices affects or favours the decentralised anomaly detection task.

Firstly, we propose an autoencoder ensemble-based method where each device of the system is provided with a 3-layer autoencoder, that is independently trained on local data. Afterwards, the trained autoencoders are collected and used to compute an outlier score as the minimum reconstruction error across the ensemble components.

Using the dataset labels to simulate different distributed settings, our solution is accurate in terms of AUC score and robust to outliers. In many experiments it outperforms equivalent centralised configurations in which all the samples are collected and more complex models are used. Moreover, our method is particularly accurate when the training data partitioning enables the devices to specialise individually on a specific normal class.

In future work, we plan to extend this model introducing incremental training phases and adaptive threshold to convert the outlier scores into binary decisions.

Acknowledgments

This work is partially funded by the following projects: Operational Knowledge from Insights and Analytics on Industrial Data (MIUR PON OK-INSAID, GA #ARS01_00917) and SoBigData++ (EU H2020, GA #871042), SAI: Social Explainable AI (EC CHIST-ERA-19-XAI-010).

References

Charu C. Aggarwal. *Outlier Analysis*. Springer International Publishing, 2017. ISBN 978-3-319-47577-6. doi: 10.1007/978-3-319-47578-3.

Charu C. Aggarwal and Saket Sathe. *Outlier ensembles: An introduction*. Springer International Publishing, apr 2017. ISBN 9783319547657. doi: 10.1007/978-3-319-54765-7.

Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof. *ACM SIGMOD Record*, 29:93–104, 6 2000. ISSN 0163-5808. doi: 10.1145/335191.335388. URL https://dl.acm.org/doi/10.1145/335191.335388.

Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey, 2019. ISSN 23318422.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection. *ACM Computing Surveys*, 41:1–58, 7 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882. URL https://dl.acm.org/doi/10.1145/1541880.1541882.

Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga. Outlier detection with autoencoder ensembles. 2017. ISBN 9781611974874. doi: 10.1137/1.9781611974973.11. Ref 93.

Yilun Jin, Xiguang Wei, Yang Liu, and Qiang Yang. Towards Utilizing Unlabeled Data in Federated Learning: A Survey and Prospective. Technical report, 2020. URL http://arxiv.org/abs/2002.11545.

Peter *et al.* Kairouz. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021. ISSN 1935-8237. doi: 10.1561/2200000083. URL http://dx.doi.org/10.1561/2200000083.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. International Conference on Learning Representations, ICLR, 12 2015. URL https://arxiv.org/abs/1412.6980v9.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Tie Luo and Sai G. Nagarajany. Distributed anomaly detection using autoencoder neural networks in WSN for IoT. Technical report, 2018.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, feb 2016. URL http://arxiv.org/abs/1602.05629.

Sutharshan Rajasegarar, Christopher Leckie, Marimuthu Palaniswami, and James C. Bezdek. Distributed anomaly detection in wireless sensor networks. In *2006 IEEE Singapore International Conference on Communication Systems, ICCS 2006*, 2006. ISBN 1424404118. doi: 10.1109/ICCS.2006.301508.

Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. *ACM International Conference Proceeding Series*, 02-Decembe: 4–11, 2014. doi: 10.1145/2689746.2689747.

Hamed Sarvari, Carlotta Domeniconi, Bardh Prenkaj, and Giovanni Stilo. Unsupervised boosting-based autoencoder ensembles for outlier detection. *arXiv*, 10 2019. URL http://arxiv.org/abs/1910.09754. https://paperswithcode.com/paper/unsupervised-boosting-based-autoencoder.

Bernhard Schölkopf, Robert C Williamson, Alexander J Smola, John Shawe-Taylor, John C Platt, et al. Support vector method for novelty detection. In *NIPS*, volume 12, pages 582–588. Citeseer, 1999.

Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. A Survey on Distributed Machine Learning. Technical Report 2, 2020.

Yan Xia, Xudong Cao, Fang Wen, Gang Hua, and Jian Sun. Learning discriminative reconstructions for unsupervised outlier removal. 2015. ISBN 9781467383912. doi: 10.1109/ICCV.2015.177. ref 45 sabokrou Adversarially¡br/¿info su dataset¡br/¿¡br/¿.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.