FEDOD: FEDERATED OUTLIER DETECTION VIA NEURAL APPROXIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Outlier detection (OD) is a crucial machine learning task with key applications in various sectors such as security, finance, and healthcare. Preserving data privacy has been increasingly important for OD due to the sensitivity of the data involved. While federated learning (FL) offers the potential to protect data privacy, it is not yet available for most classical OD algorithms, such as those based on distance and density estimation. To address this, we introduce FEDOD, the first FL-based system designed for general OD algorithms. FEDOD effectively overcomes the privacy and efficiency challenges inherent in classical OD algorithms by automatically decomposing these algorithms into a set of basic operators and approximating their behaviors using neural networks. Given the inherent compatibility of neural networks with FL, the approximated OD algorithms also become capable of privacy-preserving learning *without* data exchange. With this design, FEDOD supports over 20 popular classical OD algorithms and is readily extendable to other fields like classification and clustering. Evaluation on more than 30 benchmark and synthetic datasets demonstrates FEDOD's accuracy and efficacy over state-of-the-art baselines—compared to existing OD systems, FEDOD achieves up to $11\times$ reduction in errors and $10\times$ improvement in performance.

1 INTRODUCTION

Outlier detection (OD) is an important class of machine learning (ML) algorithms that identify observations or data points deviating from the expected behavior or patterns in a dataset Zhao et al. (2019), with numerous applications in security Khan et al. (2007), finance Lee et al. (2020), and healthcare Gupta et al. (2021). Among these applications, one key consideration is preserving data privacy, especially in scenarios where sensitive data is involved. For instance, OD has been used to identify rare diseases (as outliers) in patients, while cross-hospital data sharing is often prohibited due to regulatory reasons Pfitzner et al. (2021). This constraint restricts the analysis of an OD algorithm to data available in a single hospital, thereby constraining its performance.

To preserve user privacy while maintaining high predictive performance, *federated learning* (FL) Konečný et al. (2016) enables a new ML training paradigm, where an ML model is trained locally on a decentralized network of agents (e.g., hospitals in our motivating example). Instead of directly sending training data to a centralized server, each agent trains a model locally using private training samples. Once the local training process is completed, agents synchronize local gradients to update the model for future training. The process is repeated until achieving the desired accuracy.

FL has been widely deployed in distributed DNN training Konečný et al. (2016) and ensemble learning Smith et al. (2017). For instance, neural networks (NN; also known as "deep" methods) can conveniently learn in a distributed manner using batches, necessitating only model parameter updates without the need for data exchange Goodfellow et al. (2016). This contrasts with non-neural-network-based (also known as "classical") ML algorithms such as k nearest neighbors (kNN) and clustering Wu et al. (2008), which are thus not directly compatible with FL.

While neural-network-based OD algorithms can directly leverage FL paradigms by performing neural network computation using *local* samples Preuveneers et al. (2018); Li et al. (2020a); Gupta et al. (2021); Nguyen et al. (2019); Pei et al. (2022); Astillo et al. (2022), most classical OD algorithms, such as kNN OD Angiulli & Pizzuti (2002) and local outlier factor (LOF) Breunig et al. (2000),

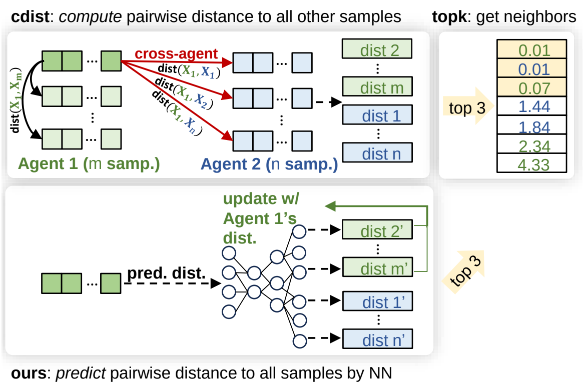


Figure 1: Demo. of dependency challenges in classical OD. *Top*: kNN OD needs to compute pairwise distances (by `cdist`) of a sample to *all* other samples (on both Agent 1 and Agent 2) and then find the top k neighbors (by `topk`). Note that cross-agent distance calculation is infeasible with privacy constraints (denoted in red arrows). *Bottom*: FEDOD uses NN to approximate `cdist` operator to *predict* pairwise distances, where the NN is updated w.r.t. the local data on Agents 1, 2, and so on. No cross-agent dist. calc. is needed.

involve *global* inter-sample data dependencies (i.e., computing the nearest neighbors of a sample requires accessing *all* other samples), and face unique challenges when training in a federated fashion.

Challenge 1: data dependency. Most classical OD algorithms have both (1) *inter-sample* data dependency, where the estimation of a sample depends on other samples , and (2) inter-feature data dependency[1], where all features contribute to the estimation of outlier scores Zhao et al. (2023). For example, kNN computes the k-th nearest neighbor of a sample as its outlier score, as shown in Fig. 1, where a larger distance indicates that the sample is far from others and thus likely to be an outlier. We need all other samples across agents to calculate and compare a sample's k-th nearest neighbor. This communication is not feasible under the FL assumptions where data reside on different agents, and cross-agent data sharing is strictly prohibited. In this example, the samples on Agent 1 do not have access to the samples on Agent 2 since the cross-agent distance calculation is infeasible; this prevents existing FL systems from supporting kNN for OD.

Challenge 2: computational cost. Classical OD algorithms involve high training and inference cost, which introduces additional difficulty under the FL setting due to the limited computational power of each agent—which is often an edge device. Given n training samples and n_t inference samples, Appx. Table A1 shows the computational complexity for many classical OD algorithms, most of which involves a complexity higher than $O(n \cdot n_t)$. High computational complexity for inference impedes the use of classical OD algorithms in most time-critical FL applications, such as real-time credit card fraud detection Boniol et al. (2021); Jiang et al. (2022; 2023).

Our approach. To address these challenges, this paper presents federated learning based outlier detection (FEDOD)[2] that can support diverse classical OD algorithms for privacy-preserving and scalable learning. Fig. 2 shows an overview of FEDOD's approach. First, we analyze a diverse group of classical OD algorithms, including distance-, density-, and tree-based algorithms, and decompose them into a small set of basic OD operators (see Fig. 2 (left) and §3). For each OD operator, we design a neural network approximation to iteratively learn the behavior of the OD operator from local data without inter-sample data dependency (see Fig. 2 (middle) and §4.1). By combining these neural networks, FEDOD approximates classical OD algorithms and leverages existing FL paradigms for privacy-preserving learning (see Fig. 2 (right) and §4.2). An example of our approach is provided in Fig. 1 bottom, where we show that FEDOD can support classical kNN OD by approximating distance calculation using a neural network to predict pair-wise distances other than calculating them. Note that the neural network here is trained only with regard to the local samples, without the need to access cross-agent samples. Our key contributions are:

- **The first FL system for diverse OD algorithms**. FEDOD supports more than 20 popular classical OD algorithms for privacy-preserving learning and can be easily extended to more.
- **Model decomposition and neural approximation**. FEDOD decomposes OD algorithms into basic operators and approximates them with neural networks with specialized local updating strategies, including k-nearest neighbors and clustering.
- **Effectiveness and scalability**. Extensive experiments on more than 30 datasets show that FEDOD outperforms the baseline with up to 11$\times$ error reduction and 10$\times$ speed up.

[1]In this work, we focus on *horizontal* FL where all agents have the same set of features and only introduce inter-feature dependency for completeness.

[2]Code is available at anonymous Google Drive: `https://tinyurl.com/fedod2023`

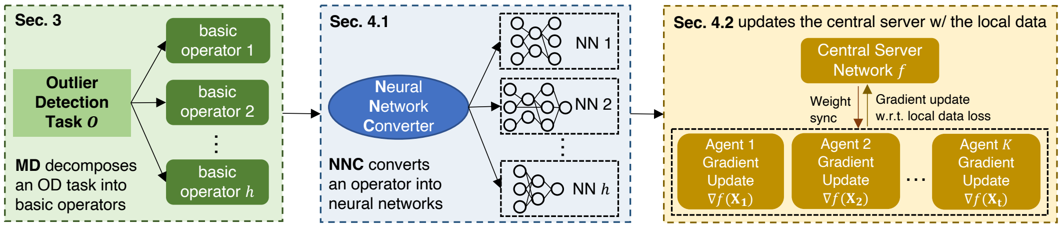

Sec. 3
Outlier Detection Task O
basic operator 1
basic operator 2
basic operator h
MD decomposes an OD task into basic operators
Sec. 4.1
Neural Network Converter
NN 1
NN 2
NN h
NNC converts an operator into neural networks
Sec. 4.2 updates the central server w/ the local data
Central Server Network f
Weight sync
Gradient update w.r.t. local data loss
Agent 1 Gradient Update $\nabla f(\mathbf{X}_1)$
Agent 2 Gradient Update $\nabla f(\mathbf{X}_2)$
Agent K Gradient Update $\nabla f(\mathbf{X}_t)$

Figure 2: FEDOD overview (§2): decomposes an OD method into shared basic operators (left; §3), which are approximated by neural nets (middle; §4.1) for privacy-preserving training (right; §4.2).

2 Federated Outlier Detection

2.1 Problem Definition

A key object of federated outlier detection is to improve the predictive performance of OD algorithms by leveraging samples across agents while preserving data privacy of each agent. Consider a horizontal FL system that supports m OD models $\mathcal{M} = \{M_1, ..., M_m\}$. It can train an OD model $M \in \mathcal{M}$ using a distributed dataset located on K agents, where each agent $k \in 1, \ldots, K$ possesses a private dataset $\mathbf{X}_k \in \mathbb{R}^{n_k \times d}$ *without* ground truth labels. Collectively, $\mathbf{X}$ is the aggregated dataset across *all* agents, where each row corresponds to an individual training sample and each column represents a feature of the sample, with a total of d unified features across all agents. The FL system outputs the outlier scores, denoted as $\mathbf{O} := M(\mathbf{X}) \in \mathbb{R}^n$, which provides anomaly scores for samples across all local agents and infer on the test samples $\mathbf{O}_{\text{test}} = M(\mathbf{X}_{\text{test}})$. The key difference between FL-based OD and general OD lies in designing privacy-preserving techniques that allow agents to train the OD model M collectively across all agents without data sharing, where general OD algorithms require access to all samples.

2.2 FedOD's Overview

To address the aforementioned challenges in data privacy and inference efficiency, a key idea behind FEDOD is to decompose a diversity of classical OD algorithms into a set of basic OD operators and use a neural network to approximate each operator. By doing so, existing FL frameworks designed for neural networks become available for classical OD algorithms through neural approximation.

Figure 2 provides an overview of FEDOD. Firstly, an OD algorithm is decomposed into low-level OD operators (§3), each of which is then converted into a neural network (§4.1). Finally, these neural networks are trained by FEDOD in a federated fashion, using specialized local update strategies (§4.2). Thus, combining these neural approximations by concatenating their network architectures emulates the outputs of the OD algorithm.

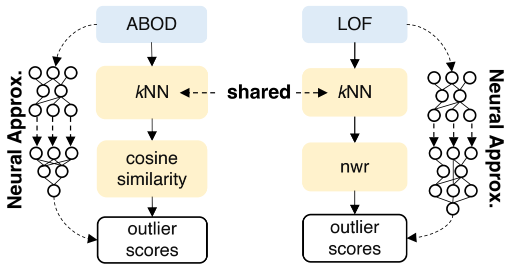


Figure 3: Example of building complex OD algorithms `ABOD` and `LOF` by *shared* basic operators, e.g., kNN. Each operator is approximated by a neural network (on two sides), which is trained sequentially as shown in the computational graph.

In addition to the kNN example in Fig. 1, Fig. 3 shows two additional examples for training `ABOD` and `LOF` in FEDOD. Specific, each complex OD algorithm is decomposed as a sequence of basic operators (e.g., `ABOD` is decomposed into kNN followed by `cosine similarity`), where each operator is approximated by a neural network in FEDOD. As illustrated in the figure, we use a combination of neural networks to approximate an OD algorithm; the output of one neural network becomes the input of the subsequent neural network—thus, these networks are trained together in an end-to-end fashion.

3 Model Decomposition

Overview. Real-world applications often require diverse OD algorithms based on the data characteristics Zhao et al. (2021; 2022); Ma et al. (2023). To support a wide range of OD algorithms

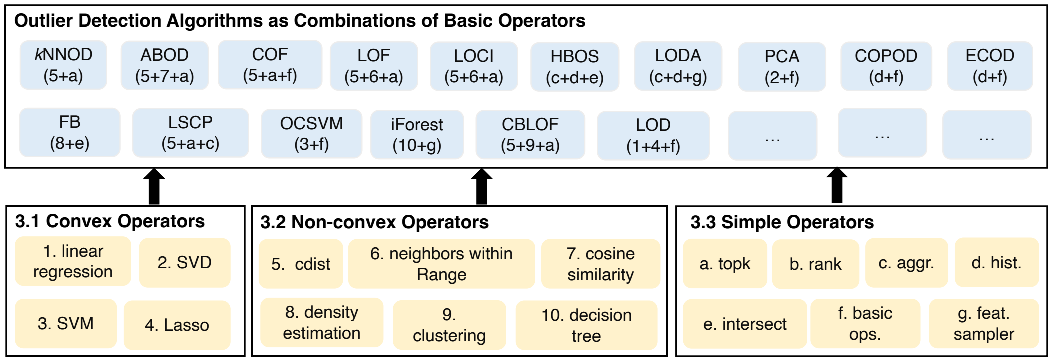


Figure 4: With model decomposition, more than 20 OD algorithms are decomposed into 4 convex operators (§3.1), 6 non-convex operators (§3.2), and 7 simple operators (§3.3). This decomposition reduces the implementation and optimization effort, and makes adding new OD operators easy.

under FL, we adapt and extend existing work on the OD decomposition model Zhao et al. (2023). The key idea is to decompose complex OD algorithms into low-level recurring operators, which are shared across diverse OD algorithms (see left of Fig. 2). For instance, the OD algorithms `ABOD` and `LOF` share a basic operator kNN to identify the k-th nearest neighbors (see Fig. 3). By identifying these recurring operators for FEDOD, we gain two benefits: (1) since each neural approximator is compatible with FL, combining them can yield FL-enabled OD algorithms, and (2) it reduces design and optimization effort—each basic operator's neural approximator only needs to be designed once.

Operator categories. After reviewing a diverse set of OD algorithms, we categorize basic OD operators into two groups: *convex operators* (§3.1) and *non-convex operators* (§3.2), depending on whether the underlying operator has a convex optimization objective. For an operator with a convex objective, its neural approximation (§4) uses a gradient-based optimization to achieve better convergence properties (e.g., fewer training epochs). In contrast, non-convex OD operators require more specialized design and pose additional challenges for convergence. Additionally, certain *simple operators* do not require any approximation (§3.3). Figure 4 illustrates the OD algorithms supported in the current implementation of FEDOD and the corresponding basic operators by category. Note that operators are independent, and FEDOD can be easily extended to support new OD operators.

3.1 CONVEX OD OPERATORS

An operator *convex* if, for every pair of points within its domain, the operator applied to the weighted average of the points is less than or equal to the weighted average of the operator applied to the points Boyd & Vandenberghe (2004). Convex OD operators offer many desirable properties. An important one is that any local minimum of a convex function is also a global minimum, which greatly simplifies the search for a global optimal. Thus, using a gradient-based optimization in a neural network can generally approximate convex operators well via gradient-based optimizations Snyman et al. (2005) and takes less training time. Fig. 4 and Appx. B.1 left discuss a few commonly used convex operators by classical OD algorithms with additional ones including support vector machine (SVM) Tibshirani (1996) and Lasso Cortes & Vapnik (1995).

3.2 NON-CONVEX OD OPERATORS

A non-convex operator is one that does not satisfy the convex condition. Handling non-convex operators in optimization is challenging because they often have multiple local minima, and standard gradient-based optimization methods may get stuck in these local minima rather than finding the global minimum, leading to more complexity of neural approximation with gradient-based optimization Boyd & Vandenberghe (2004). Fig. 4 (middle) and Appx. B.2 shows some key operators supported in FEDOD in addition to cosine similarity calculation, and density estimation.

3.3 SIMPLE OD OPERATORS

Not all operators necessitate a (neural) approximation. Certain operators do not involve computation and/or optimization, and can thus be directly executed federated. This category includes most sorting and ranking operators. For instance, the kNN operator comprises two primary steps: (1) computing the pairwise distances among samples, and (2) identifying the top k distances per sample. The first step involves computation and should be approximated by a neural network, while the second step, which is about finding the top k values, can be directly applied to the output of the first step, without the need for any approximation. Figure 4 (right) summarizes the simple operators in FEDOD.

4 OD APPROXIMATION

4.1 NEURAL NETWORK CONVERTER

As existing FL frameworks are designed for neural networks Li et al. (2020a), we propose converting basic OD operators (see §3) each into a neural network. This conversion is theoretically feasible since a neural network is a "universal approximator" Hornik et al. (1989). The choice of neural network approximations, such as architectures, can be flexible, provided that the model capacity is sufficient — the neural networks are adequately complex for approximation (see §5.4.1). Under this criterion, fully-connected multi-layer perceptrons (MLP) Rumelhart et al. (1986) serve as the default choice in FEDOD. We have also explored more recent architectures like Transformers Vaswani et al. (2017) (see §5.4.2). FEDOD's *neural network converter* (NNC) translates basic OD operators in Fig. 4 into neural networks, as depicted in the middle of Fig. 2. Notably, this is the first proposal for neural approximations of general classical OD algorithms such as kNN and clustering. With the OD model decomposition and the NNC, a variety of classical OD algorithms can be decomposed into and approximated by neural networks, which can then be trained in existing FL systems.

4.2 TRAINING METHODOLOGY

Local ground truth. Even after OD operators are converted to neural networks, training them on each local agent without data sharing requires special considerations. As depicted on the right of Fig. 2, FEDOD's goal is to train a central model $f(\cdot)$, i.e., the neural network generated from NNC as an approximation of the underlying OD operator.

To update $f(\cdot)$, we can average local gradients using the data on each agent, i.e., $\nabla f(\mathbf{X}_1)$, $\nabla f(\mathbf{X}_2)$, a technique known as Federated Averaging (FedAvg)[3] in Eq. (2) McMahan et al. (2017). However, local gradients are not directly computable due to global data dependencies (i.e., the global ground truth of k-th nearest neighbors is unavailable).

In FEDOD, we introduce a set of novel *local-update strategies* for each neural OD operator. The key is to design a loss function that allows each neural OD operator to update the model with respect to the *local* data only, to approximate the *global* ground truth.

Clustering example. The goal of k-means clustering is to partition n samples across all agents into clusters defined by sample similarity. As described in §3.2, the sample similarity can be measured by pairwise Euclidean distances —samples in the same cluster should be close. For an $n \times d$ input matrix, all n samples are needed for the Euclidean distance calculation (i.e., `cdist`). However, in the FL setting, the i-th agent can only access private data $\mathbf{X}_i \in \mathbb{R}^{n_k \times d}$ and the local distance of $\mathbf{D}_i = \text{cdist}(\mathbf{X}_i, \mathbf{X}_i)$. We cannot access the global sample similarity across all agents by only looking at a partition of data on a local agent.

Suppose we use a neural network to approximate k-Means without considering privacy preservation, the model may minimize the discrepancy between the predicted cluster assignment and the ground truth cluster assignment (unavailable in our settings). Differently, FEDOD performs local updates without accessing the global ground truth of cluster assignments, as shown in Algorithm 1. The key is to design a loss function *solely based on the local data*.

First, we initialize the central model $f(\cdot)$ with parameters $\mathbf{w}$ and pre-compute the local pairwise distances for all K agents, i.e., $\mathbf{D}_k = \text{cdist}(\mathbf{X}_k, \mathbf{X}_k)$ (lines 1-4). Second, for the k-th agent, let $\widehat{\mathbf{c}} = f_k(\mathbf{X}_k)$ denote the predicted cluster labels for the local data by the current model. Given there are n_k samples on the k-th agent, our loss defined in Eq. 1 aims to minimize the intra-cluster distance and maximize the inter-cluster distance for each local sample given the predicted cluster labels by $f(\cdot)$ (lines 5-10). Note this only uses the data on the k-th agent.

[3]Other FL frameworks can also be applied; we use FedAvg as an example.

Algorithm 1 Neural approximation of k-Means clustering

Input: Central neural model $f(\cdot)$ with weights $\mathbf{w}$ and objective function $\mathcal{L}$, K local agents where the k-th agent with private data $\mathbf{X}_k \in \mathbb{R}^{n_k \times d}$ in the same d-dimension feature space; local neural network models $\{f_1, \ldots, f_K\}$; training budget T

Output: The trained neural model $f(\cdot)$ across local agents

```
 1: Initialize the central model $f(\cdot)$
 2: for each local agent $k = 1$ to $K$ do
 3:     Compute pairwise local distance matrix $\mathbf{D}_k = \text{cdist}(\mathbf{X}_k, \mathbf{X}_k)$
 4: end for
 5: for $t = 1$ to $T$ do
 6:     for each local agent $k = 1$ to $K$ do
 7:         Get current central model weights $\mathbf{w}^t$ and initialize the local model $f_k$ to it
 8:         Predict the cluster labels on the local data $\widehat{\mathbf{c}} = f_k(\mathbf{X}_k)$
 9:         Compute local loss $\mathcal{L}_k = (\widehat{\mathbf{c}}, \mathbf{D}_k)$ and gradient $\nabla \mathbf{w}_k^{(t)}$ to minimize the intra-cluster and
            maximize the inter-cluster distance given the predicted cluster labels by Eq. (1)
10:     end for
11:     Update the central model $\mathbf{w}^{(t+1)}$ by FedAvg in Eq. (2)
12: end for
```

$$\mathcal{L}(\widehat{\mathbf{c}}, \mathbf{D}_k)) = \sum_{i=1}^{n_k} \sum_{j \neq i} (\underbrace{- \sum_{\widehat{\mathbf{c}}_i = \widehat{\mathbf{c}}_j} \mathbf{D}_{i,j}}_{\text{min. intra dist.}} + \underbrace{\sum_{\widehat{\mathbf{c}}_i \neq \widehat{\mathbf{c}}_j} \mathbf{D}_{i,j}}_{\text{max. inter dist.}}) \tag{1}$$

At t-th iteration, FEDOD iterates over all the agents to aggregate the local gradients $\mathbf{w}_1^{(t)}, \ldots, \mathbf{w}_K^{(t)}$ (line 11).; one straightforward way is to use FedAvg as shown in Eq. (2).

$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} + \frac{1}{n} \sum_{k=1}^{K} n_k \cdot \nabla \mathbf{w}_k^{(t)} \tag{2}$$

In this way, we could update the central model without accessing any global samples as well as ground truth labels. We design a local loss function For each supported operator in FEDOD.

4.3 ADVANTAGES OF FEDOD

Anytime inference. An anytime algorithm can return a valid solution even if it is not fully complete Cutkosky (2019), which is particularly useful in real-time OD applications. These algorithms typically present a trade-off between computational resources, like time or memory, and the quality of the solution; for example, longer training time results in better accuracy. Neural networks are naturally anytime algorithms due to their iterative training nature—we can make predictions at any time during the training process. In contrast, most classical OD algorithms (see Table A1) cannot make predictions until the training is fully complete, limiting their applicability in time-critical applications. However, FEDOD's neural approximation techniques convert classical OD algorithms to anytime algorithms, thus enhancing their flexibility.

Fast inference and optimization techniques. Existing OD algorithms often have high inference cost due to distance calculation and/or density estimation. By employing neural approximation in FEDOD, the inference time can be largely reduced to a single forward pass with the data. Thus, for large datasets with high feature dimensions, FEDOD offers shorter inference time. Additionally, any neural network optimization techniques, such as parallelization Schneider et al. (2021); Wang et al. (2023), quantization Zhao et al. (2023), also apply to shallow OD algorithms via FEDOD. See §5.3 for a detailed comparison of inference times.

Generality and extensibility. Thanks to the generality of basic operators in FEDOD, it can be easily extended to support ML algorithms beyond OD, such like classification. Moreover, integrating new operators into FEDOD is straightforward, requiring just two simple steps. First, we need to understand the general properties of the operator to be added, including its convexity, inputs, and outputs. Second, we need to design a local loss function that can be computed solely based on the local data, as described in §4.2. Our experience of developing FEDOD shows that it generally takes a few hours to add a new OD operator and its neural approximation.

5 EXPERIMENTS

Our experiments aim to answer the following questions: (i) How does the detection performance of FEDOD compare to baselines, both with and without privacy preservation? (§5.2) (ii) How efficient and scalable is FEDOD when handling larger datasets? (§5.3) (iii) How do the designs in FEDOD impact its effectiveness? (§5.4)

5.1 EXPERIMENT SETUP

Datasets. Appx Table C2 displays over 21 real-world OD datasets used in this study, primarily sourced from two popular repositories, i.e., DAMI Campos et al. (2016) and ODDS Rayana (2016). These datasets have been widely employed in OD research Tran et al. (2020); Schmidl et al. (2022); Han et al. (2022); Zhao et al. (2023).

OD algorithms and operators. We include five diverse OD algorithms to show the effectiveness of FEDOD: (1) Distance-based kNN for OD Ramaswamy et al. (2000); (2) Density-based local outlier factor (LOF) Breunig et al. (2000); (3) Linear PCA for OD Shyu et al. (2003); (4) Clustering-based OD method (CBLOF) He et al. (2003); and (5) Tree-based isolation forest (iForest) Liu et al. (2008).

Implementation and environment. FEDOD is implemented on top of PyTorch Paszke et al. (2019), where most algorithms only depend on 2-3 operators with low complexity. All the experiments are performed on an Amazon EC2 p3.2xlarge cluster with an Intel Xeon CPU, 61GB DRAM, and an NVIDIA Tesla V100 GPU with 16GB RAM.

Baselines. As the first work for privacy-preserving OD with shallow methods, FEDOD is compared with (1) *ground truth* where the access to all samples is assumed (no privacy preservation at all); we use PyOD Zhao et al. (2019) to get the results and (2) *direct* results where we train K individual models (one per local dataset) and concatenate the results (data privacy is preserved).

Evaluation metrics. Enabled by model approximation, FEDOD's goal is to achieve similar performance as the *ground truth* baseline while preserving privacy. With that in mind, we measure the performance difference[4] to the ground truth—the smaller the difference, the better the method.

5.2 END-TO-END COMPARISONS

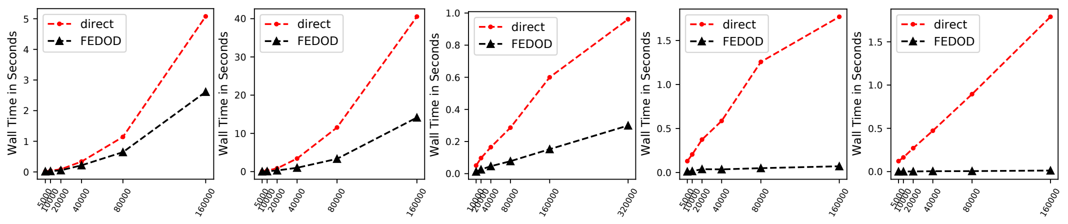
Appx. Table C3 illustrates that FEDOD achieves performance close to the ground truth and surpasses the *direct* baseline across all five OD algorithms with differing characteristics. Specifically, FEDOD displays less than 5% ROC-AUC difference from the ground truth across all five OD algorithms (1.79%, 3.93%, 2.05%, 1.96%, and 4.70% for kNN, LOF, PCA, CBLOF, and iForest, respectively), whereas the *direct* method shows up to a 19% performance difference. It is worth noting that FEDOD is 11 times better than direct on kNN on average (1.79% vs. 18.92%).

Convex operators seem to exhibit smaller approximation differences. As discussed in §3.1, PCA (Table C3c) presents relatively minor performance differences (2.05%) and variations (2.44%). This can be attributed to the favorable convergence properties of convex operators, whereas non-convex operators like LOF (Table C3b, 3.93% $\pm$ 4.17%) and iForest (Table C3e, 4.70%$\pm$ 4.72%) may present much larger differences and variations (which are less desirable).

5.3 SCALABLITY OF FEDOD

We evaluate the scalability of FEDOD across datasets of varying sizes (ranging from 5,000 to 320,000 samples with 1,000 features), thereby simulating scenarios of OD on high-dimensional, large datasets. Fig. 5 illustrates the inference time of FEDOD (in black) and *direct* baseline (in red).

[4]We use the area under the Receiver Operating Characteristic curve (ROC) as the performance; this can be substituted with any other measure of interest.



(a) kNN (b) LOF (c) PCA (d) CBLOF (e) iForest

Figure 5: Scalability plot of algorithms in FEDOD, where it scales well with increasing samples. FEDOD exhibits superior efficiency compared to *direct* across all datasets, particularly larger ones. For example, FEDOD achieves a $10\times$ speed-up over the direct method on the `CBLOF` and `iForest` datasets. One of the primary reasons for this improvement is that the *direct* method needs to invoke multiple models for inference, while FEDOD relies solely on a single central model. Moreover, FEDOD only requires a single pass of the neural network, which is significantly more economical than the *direct* method, which relies on distance and/or density estimation.

FEDOD demonstrates robust scalability with larger datasets. As shown in Fig. 5, the inference time scales linearly about the number of inference samples. This is attributed to the fact that the network parameters are fixed—the inference time is determined solely by the number of inference samples.

5.4 Ablation Studies and Additional Analysis

5.4.1 Effect of Model Capacity

We assess model capacity by modifying the number of hidden neurons (x-axis) and the number of layers (y-axis) in the neural approximation of kNN, and compare their performance variance with the *ground truth* in Fig. 6. The results indicate that FEDOD's performance is fairly insensitive to the neural model capacity, provided it is adequately large, with all variations remaining under 3%. Thus, we employ 64 hidden neurons and 2 layers across all experiments. Utilizing a comparatively small model curbs both training and inference costs. Also note that the basic operators are generally simple, and thus small neural network could be more cost-effective.

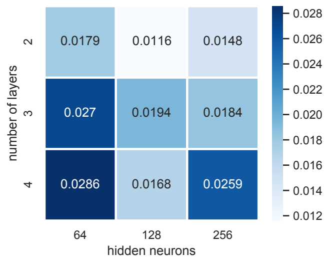


Figure 6: Ablation studies on model capacity (i.e., size of neural networks) of approximation: (x-axis) the number of hidden neurons and (y-axis) the number of layers. We show the avg. performance diff. (smaller the better) across all datasets on kNN (Table C3a); FEDOD is insensitive model capacity with small differences with varying sizes of neural networks.

5.4.2 The Choice of Backbone in FedOD

We evaluate the performance of using a simple MLP versus the more recent transformers Vaswani et al. (2017) as the backbone of the neural approximator in §4. Transformers, capable of capturing attention between both data points and features, have demonstrated strong performance in recent ADBench studies Han et al. (2022). This might help us establish a relationship that correlates input directly to output, based not only on feature combinations but also on other data points. Here we utilize a transformer model with two self-attention heads and a stacking depth of two.

Appx. Table C3f presents the performance of kNN using both MLP (Table C3a) and transformers (Table C3f). We observe that MLP outperforms transformers in 14 out of 21 datasets, while the average performances are comparable (1.79% vs. 2.77%). Note that both are superior to the *direct* baselines (18.92%). One reason for the lower performance of transformers might be their complexity and sensitivity to hyperparameters Chen et al. (2020). This affirms the choice of MLP as the default backbone for FEDOD, given its simplicity, speed, and robustness.

6 Limitations and Future Directions

Automated network conversion. At present, the conversion of operators in FEDOD is carried out manually, a process that could be further automated using meta-learning Vanschoren (2018).

For instance, we could train multiple neural approximators on existing datasets to evaluate their performance. When presented with a new dataset, we identify the most similar historical dataset and transfer the optimal neural configurations Zhao et al. (2021) in a zero-shot manner. Also, network parameters from similar historical datasets could be transferred Scott et al. (2018), thus reducing the training cost.

Inference optimization. On top of the already efficient inference, FEDOD may further leverage acceleration techniques for neural networks, including quantization Hubara et al. (2017), distributed and multi-GPU learning Jia et al. (2017), etc. However, it is worth noting that further acceleration may be at the cost of larger performance differences.

7 RELATED WORK

Federated learning for OD. OD is a crucial task in a variety of applications such as credit card fraud detection Zhong et al. (2020), cybersecurity Mothukuri et al. (2021), healthcare Gupta et al. (2021), and environmental anomaly detection Chandola et al. (2009). In many circumstances, due to privacy constraints or regulatory requirements, data cannot be shared, rendering traditional OD methods untenable. There exists a huge need for privacy-preserving OD methods that can operate without sharing data, thereby preventing potential misuse of sensitive info Sater & Hamza (2021).

Federated learning (FL), as a leading privacy-preserving framework, has been applied to various *deep* OD algorithms. These algorithms employ different neural architectures, including recurrent neural networks (RNN) Nguyen et al. (2019); Mothukuri et al. (2021), autoencoders Pei et al. (2022), LSTM Sater & Hamza (2021), and CNN Astillo et al. (2022). There are OD applications in diverse domains such as healthcare Gupta et al. (2021); Astillo et al. (2022), internet of things (IoT) Nguyen et al. (2019); Mothukuri et al. (2021), network security Pei et al. (2022), and smart buildings Sater & Hamza (2021). Most of these works primarily focus on horizontal FL, where different agents (e.g., mobile devices or workstations) hold a partition of samples from the same feature space. FL for deep OD typically follows the following procedures Sater & Hamza (2021). Initially, a global neural network OD model is initialized. Then, local datasets are utilized to update the model parameters through local training using techniques such as stochastic gradient descent (SGD). Finally, the updated model parameters from each device are aggregated to update the global model for OD, such as Federated Averaging (FedAvg) McMahan et al. (2017).

However, most classical OD algorithms cannot leverage existing FL frameworks due to inter-sample data dependency as we elaborate in the introduction. Table A1 summarizes a diverse group of classical OD algorithms, many of which lack a straightforward FL solution due to data dependency. Hence, FEDOD is designed to address this gap while accelerating inference.

Neural approximation for OD. Neural networks are widely recognized for their remarkable function approximation capabilities Goodfellow et al. (2016). This is predominantly due to their ability to model complex, high-dimensional, and nonlinear relationships inherent in data. The Universal Approximation Theorem provides theoretical support to this notion by asserting that a feedforward network with a single hidden layer containing a finite number of neurons can approximate continuous functions under specific conditions Hornik et al. (1989). Also, deep neural networks can represent complex functions more compactly. This allows them to model complex patterns in data, making them useful in various applications such as language translation and autonomous driving LeCun et al. (2015). In FEDOD, we employ neural networks to approximate OD operators for the dual purposes of *preserving privacy* and *accelerating inference*.

8 CONCLUSION

We introduce FEDOD, a novel system designed to overcome challenges associated with privacy preservation and efficiency in outlier detection applications. FEDOD enables popular federated learning paradigm and extends its benefits to over 20 shallow (non-neural-network) OD algorithms. The key steps include decomposing OD algorithms into shared operators for neural approximation, after which federated learning becomes compatible with the operators. Through extensive experiments on more than 30 datasets and 5 diverse detection algorithms, we demonstrate that FEDOD can efficiently approximate OD algorithms while ensuring privacy. Future work can extend FEDOD's support to more other ML tasks and optimize it by neural network acceleration techniques.

REFERENCES

Charu C. Aggarwal. *Outlier Analysis*. Springer, 2013.

Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen (eds.), *Principles of Data Mining and Knowledge Discovery, 6th European Conference, PKDD 2002, Helsinki, Finland, August 19-23, 2002, Proceedings*, volume 2431 of *Lecture Notes in Computer Science*, pp. 15–26. Springer, 2002. doi: 10.1007/3-540-45681-3_2. URL `https://doi.org/10.1007/3-540-45681-3_2`.

Philip Virgil Astillo, Daniel Gerbi Duguma, Hoonyong Park, Jiyoon Kim, Bonam Kim, and Ilsun You. Federated intelligence of anomaly detection agent in iotmd-enabled diabetes management control system. *Future Generation Computer Systems*, 128:395–405, 2022.

Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J Franklin. Sand: streaming subsequence anomaly detection. *VLDB*, 14(10):1717–1729, 2021.

Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. In *SIGMOD*, pp. 93–104. ACM, 2000. ISBN 1-58113-217-4.

Guilherme O Campos, Arthur Zimek, Jörg Sander, Ricardo JGB Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E Houle. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data mining and knowledge discovery*, 30 (4):891–927, 2016.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33:15834–15846, 2020.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

Ashok Cutkosky. Anytime online-to-batch, optimism and acceleration. In *International conference on machine learning*, pp. 1446–1454. PMLR, 2019.

Evelyn Fix and Joseph Lawson Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, 57 (3):238–247, 1989.

Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI*, pp. 59–63, 2012.

Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT, 2016.

Deepti Gupta, Olumide Kayode, Smriti Bhatt, Maanak Gupta, and Ali Saman Tosun. Hierarchical federated learning based anomaly detection using digital twins for smart healthcare. In *2021 IEEE 7th International Conference on Collaboration and Internet Computing (CIC)*, pp. 16–25. IEEE, 2021.

Songqiao Han, Xiyang Hu, Hailiang Huang, Mingqi Jiang, and Yue Zhao. Adbench: Anomaly detection benchmark. *arXiv preprint arXiv:2206.09426*, 2022.

Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern recognition letters*, 24(9-10):1641–1650, 2003.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

Zhihao Jia, Yongkee Kwon, Galen Shipman, Pat McCormick, Mattan Erez, and Alex Aiken. A distributed multi-gpu system for fast graph processing. *Proceedings of the VLDB Endowment*, 11 (3):297–310, 2017.

Jiaxin Jiang, Yuan Li, Bingsheng He, Bryan Hooi, Jia Chen, and Johan Kok Zhi Kang. Spade: A real-time fraud detection framework on evolving graphs. *Proceedings of the VLDB Endowment*, 16(3):461–469, 2022.

Minqi Jiang, Chaochuan Hou, Ao Zheng, Songqiao Han, Hailiang Huang, Qingsong Wen, Xiyang Hu, and Yue Zhao. Adgym: Design choices for deep anomaly detection. *Advances in Neural Information Processing Systems*, 36, 2023.

Latifur Khan, Mamoun Awad, and Bhavani Thuraisingham. A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB journal*, 16:507–521, 2007.

Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Outlier detection in axis-parallel subspaces of high dimensional data. In *Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 Proceedings 13*, pp. 831–838. Springer, 2009.

Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17, 1976.

Aleksandar Lazarevic and Vipin Kumar. Feature bagging for outlier detection. In *KDD*, pp. 157–166. ACM, 2005.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Meng-Chieh Lee, Yue Zhao, Aluna Wang, Pierre Jinghong Liang, Leman Akoglu, Vincent S. Tseng, and Christos Faloutsos. Autoaudit: Mining accounting and time-evolving graphs. In *Big Data*, pp. 950–956. IEEE, 2020.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020a.

Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. COPOD: copula-based outlier detection. In *ICDM*, pp. 1118–1123. IEEE, 2020b.

Zheng Li, Yue Zhao, Xiyang Hu, Nicola Botta, Cezar Ionescu, and George Chen. Ecod: Unsupervised outlier detection using empirical cumulative distribution functions. *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2022. doi: 10.1109/TKDE.2022.3159580.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *ICDM*, pp. 413–422. IEEE Computer Society, 2008.

Martin Q Ma, Yue Zhao, Xiaorong Zhang, and Leman Akoglu. The need for unsupervised outlier model selection: A review and evaluation of internal evaluation strategies. *ACM SIGKDD Explorations Newsletter*, 25(1), 2023.

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pp. 281–297. Oakland, CA, USA, 1967.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Viraaji Mothukuri, Prachi Khare, Reza M Parizi, Seyedamin Pouriyeh, Ali Dehghantanha, and Gautam Srivastava. Federated-learning-based anomaly detection for iot security attacks. *IEEE Internet of Things Journal*, 9(4):2545–2554, 2021.

Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N Asokan, and Ahmad-Reza Sadeghi. Dïot: A federated self-learning anomaly detection system for iot. In *2019 IEEE 39th International conference on distributed computing systems (ICDCS)*, pp. 756–767. IEEE, 2019.

Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Proceedings 19th international conference on data engineering*, pp. 315–326. IEEE, 2003.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeuIPS*, 32:8026–8037, 2019.

Jiaming Pei, Kaiyang Zhong, Mian Ahmad Jan, and Jinhai Li. Personalized federated learning framework for network traffic anomaly detection. *Computer Networks*, 209:108906, 2022.

Tomás Pevný. Loda: Lightweight on-line detector of anomalies. *Mach. Learn.*, 102(2):275–304, 2016.

Bjarne Pfitzner, Nico Steckhan, and Bert Arnrich. Federated learning in a medical context: a systematic literature review. *ACM Transactions on Internet Technology (TOIT)*, 21(2):1–31, 2021.

Davy Preuveneers, Vera Rimmer, Ilias Tsingenopoulos, Jan Spooren, Wouter Joosen, and Elisabeth Ilie-Zudor. Chained anomaly detection models for federated learning: An intrusion detection case study. *Applied Sciences*, 8(12):2663, 2018.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 427–438, 2000.

Shebuti Rayana. ODDS library, 2016. URL `https://odds.cs.stonybrook.edu`.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Raed Abdel Sater and A Ben Hamza. A federated learning approach to anomaly detection in smart buildings. *ACM Transactions on Internet of Things*, 2(4):1–23, 2021.

Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: a comprehensive evaluation. *Proceedings of the VLDB Endowment*, 15(9):1779–1797, 2022.

Johannes Schneider, Phillip Wenig, and Thorsten Papenbrock. Distributed detection of sequential anomalies in univariate time series. *The VLDB Journal*, 30(4):579–602, 2021.

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

Tyler Scott, Karl Ridgeway, and Michael C Mozer. Adapted deep embeddings: A synthesis of methods for k-shot inductive transfer learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. A novel anomaly detection scheme based on principal component classifier. Technical report, 2003.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017.

Jan A Snyman, Daniel N Wilke, et al. *Practical mathematical optimization*. Springer, 2005.

Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David W Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Advances in Knowledge Discovery and Data Mining: 6th Pacific-Asia Conference, PAKDD 2002 Taipei, Taiwan, May 6–8, 2002 Proceedings 6*, pp. 535–548. Springer, 2002.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

Luan Tran, Min Y Mun, and Cyrus Shahabi. Real-time distance-based outlier detection in data streams. *VLDB*, 14(2):141–153, 2020.

Joaquin Vanschoren. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Weiyang Wang, Moein Khazraee, Zhizhen Zhong, Manya Ghobadi, Zhihao Jia, Dheevatsa Mudigere, Ying Zhang, and Anthony Kewitsch. {TopoOpt}: Co-optimizing network topology and parallelization strategy for distributed training jobs. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pp. 739–767, 2023.

Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, Philip S Yu, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14:1–37, 2008.

Yue Zhao, Zain Nasrullah, and Zheng Li. PyOD: A python toolbox for scalable outlier detection. *JMLR*, 20:96:1–96:7, 2019.

Yue Zhao, Ryan Rossi, and Leman Akoglu. Automatic unsupervised outlier model selection. *Advances in Neural Information Processing Systems*, 34:4489–4502, 2021.

Yue Zhao, Sean Zhang, and Leman Akoglu. Toward unsupervised outlier model selection. In *2022 IEEE International Conference on Data Mining (ICDM)*, pp. 773–782. IEEE, 2022.

Yue Zhao, George H. Chen, and Zhihao Jia. Tod: Gpu-accelerated outlier detection via tensor operations. *Proceedings of the VLDB Endowment*, 16(3), 2023.

Qiwei Zhong, Yang Liu, Xiang Ao, Binbin Hu, Jinghua Feng, Jiayu Tang, and Qing He. Financial defaulter detection on online credit payment via multi-view attributed heterogeneous information network. In *WWW*, pp. 785–795. ACM, 2020.

Appendix

A Classicial OD Algorithms

Classical OD algorithms involve high training and inference cost, which introduces additional difficulty under the FL setting due to each agent's limited computational power, which is often an edge device. Given n training samples and n_t inference samples, Appx. Table A1 shows the computational complexity for many classical OD algorithms, most of which involve a complexity higher than $O(n \cdot n_t)$.

Table A1: Key classical OD algorithms and whether they are inter-sample dependent (i.e., requiring other samples for computation) and inter-feature dependent (i.e., requiring other features for computation). These methods are generally costly in inference, where n and n_t denote the numbers of training and inference samples, and d is the number of features. Ensemble algorithms' inference time depends on base estimators' complexity (the last three rows).

Algorithm	**Inter-sample**	**Inter-feature**	**Infer time**
kNN (OD) Ramaswamy et al. (2000)	✓	✓	$n \cdot n_t \cdot d$
COF Tang et al. (2002)	✓	✓	$n \cdot n_t \cdot d$
LOF Breunig et al. (2000)	✓	✓	$n \cdot n_t \cdot d$
LOCI Papadimitriou et al. (2003)	✓	✓	$n \cdot n_t \cdot d$
SOD Kriegel et al. (2009)	✓	✓	$n \cdot n_t \cdot d$
CBLOF He et al. (2003)	✓	✓	$n \cdot n_t \cdot d$
HBOS Goldstein & Dengel (2012)	✓	✗	$n_t \cdot d$
COPOD Li et al. (2020b)	✓	✗	$n_t \cdot d$
ECOD Li et al. (2022)	✓	✗	$n_t \cdot d$
PCA Shyu et al. (2003)	✓	✓	$n_t \cdot d$
OCSVM Schölkopf et al. (2001)	✓	✓	$n_t \cdot d$
LODA Pevný (2016)	✗	✗	N/A
FB Lazarevic & Kumar (2005)	✓	✗	N/A
iForest Liu et al. (2008)	✗	✗	N/A

B Details on Model Decomposition

B.1 Convex Operators

Linear regression is a popular statistical method that investigates the linear relationship between predictor variables James et al. (2013), which can be optimized by gradient descent via minimizing a convex mean squared loss. For OD, it can identify outlying samples by examining the residuals Aggarwal (2013), suggesting that these data points do not follow the same pattern as most of the data.

Singular value decomposition (SVD) is a matrix decomposition technique used extensively in ML Golub & Van Loan (2013). Notably, SVD can be transformed into convex problems for solving Boyd & Vandenberghe (2004). Under the context of OD, SVD is an important part of *principal component analysis* (PCA), which identifies outliers by projecting the data onto the subspace spanned by the leading singular vectors; points far away from this subspace's origin can be considered outliers Aggarwal (2013).

B.2 Non-convex Operators

kNN is a non-parametric, instance-based ML method Fix & Hodges (1989). The central concept is that similar data samples should yield similar outputs. Sample similarity is often gauged by distances, which does not constitute a convex problem since it is not an optimization task.

Clustering endeavors to partition samples into a predefined number of clusters based on certain criteria, e.g., proximity to the cluster center. For instance, the popular k-Means method MacQueen et al. (1967) is not a convex problem, as its objective is the sum of squared distances from each point to the centroid of its assigned cluster.

Decision tree is a tree-like model of decisions, where each internal node corresponds to a feature in the data, each branch signifies a decision rule, and each leaf node represents an outcome Quinlan (1986). It is known that the problem of finding an optimal decision tree is NP-complete and non-convex, with a combinatorial explosion of potential tree structures (and hence many local optima) as the number of features increases Laurent & Rivest (1976).

C Experiment Details

C.1 Dataset and Code

Appx Table C2 displays over 21 real-world OD datasets used in this study, primarily sourced from two popular repositories, i.e., DAMI Campos et al. (2016) and ODDS Rayana (2016). These datasets have been widely employed in OD research Tran et al. (2020); Schmidl et al. (2022); Han et al. (2022); Zhao et al. (2023).

Table C2: Twenty-one real-world OD datasets used in the experiments. We also create and use synthetic datasets throughout the experiments to demonstrate the results on larger datasets.

Dataset	**Source**	**#Samples**	**#Dims**	**%Outlier**
Annthyroid	DAMI	7129	21	7.49
Cardio	DAMI	2114	21	22.04
Glass	DAMI	214	7	4.21
Heart	DAMI	270	13	44.44
InternetAds	DAMI	1966	1555	18.72
PageBlocks	DAMI	5393	10	9.46
Pima	DAMI	768	7	34.9
SpamBase	DAMI	4207	57	39.91
Stamps	DAMI	340	9	9.12
WBC	DAMI	223	9	4.48
ionosphere	ODDS	351	33	35.9
mammog.	ODDS	11183	6	2.32
mnist	ODDS	7603	100	9.21
pima	ODDS	768	8	34.9
satellite	ODDS	6435	36	31.64
satimage-2	ODDS	5803	36	1.22
speech	ODDS	3686	400	1.65
thyroid	ODDS	3772	6	2.47
vowels	ODDS	1456	12	3.43
wbc	ODDS	378	30	5.56

The demonstration code is available at anonymous Google Drive: `https://tinyurl.com/fedod2023`.

C.2 End-to-End Results

Appx. Table C3 illustrates that FedOD achieves performance close to the ground truth and surpasses the *direct* baseline across all five OD algorithms with differing characteristics. Specifically, FedOD displays less than 5% ROC-AUC difference from the ground truth across all five OD algorithms (1.79%, 3.93%, 2.05%, 1.96%, and 4.70% for kNN, LOF, PCA, CBLOF, and iForest, respectively), whereas the *direct* method shows up to a 19% performance difference. It is worth noting that FedOD is 11 times better than direct on kNN on average (1.79% vs. 18.92%).

Table C3: ROC-AUC comparison of *ground truth*, *ours*, and *direct* across five diverse OD algorithms (subtable C3a to C3e). Across all five OD methods, *ours* consistently shows competitive performance with small differences to *ground truth* (up to 5% on average), outperforming the *direct* method in most cases. By maintaining a lower Avg. $|\Delta|$ values to the ground truth (up to $11\times$ better than *direct*), FEDOD demonstrates its efficacy and robustness across OD scenarios. Subtable C3f shows the ablation study on using MLP (Ours) vs. transformers (Transf.) as the backbone of FEDOD; see more detailed analysis in §5.4.2.

Dataset	Ground truth	Ours	ΔOurs	Direct	ΔDirect
Annthyroid	0.6616	0.6188	-6.47%	0.5389	-18.54%
Cardio	0.4833	0.4923	1.87%	0.6503	34.56%
Glass	0.7619	0.7381	-3.12%	0.4762	-37.50%
Heart	0.5628	0.5411	-3.85%	0.4776	-15.13%
InternetAds	0.6868	0.6770	-1.44%	0.6623	-3.57%
PageBlocks	0.9128	0.9164	0.39%	0.8083	-11.45%
Pima	0.7734	0.7831	1.26%	0.6817	-11.85%
SpamBase	0.5897	0.5938	0.70%	0.5358	-9.13%
Stamps	0.8857	0.9111	2.87%	0.4508	-49.10%
WBC	0.9773	0.9773	0.00%	1.0000	2.33%
arrhythmia	0.8017	0.8006	-0.13%	0.8439	5.26%
ionosphere	0.9294	0.9145	-1.61%	0.6728	-27.61%
mammog.	0.8505	0.8807	3.56%	0.6484	-23.77%
mnist	0.8462	0.8544	0.97%	0.7249	-14.33%
pima	0.7372	0.7497	1.69%	0.6534	-11.36%
satellite	0.6879	0.7167	4.19%	0.4711	-31.52%
satimage-2	0.9826	0.9966	1.42%	0.9420	-4.14%
speech	0.5750	0.5678	-1.25%	0.6266	8.99%
thyroid	0.9720	0.9677	-0.43%	0.8454	-13.02%
vowels	0.9731	0.9735	0.03%	0.4555	-53.19%
wbc	0.9310	0.9333	0.26%	0.8286	-11.00%
Avg. $\|\Delta\|$	N/A	1.79%± 1.67%		18.92%± 14.73%	

(a) kNN (distance-based)

Dataset	Ground truth	Ours	ΔOurs	Direct	ΔDirect
Annthyroid	0.6737	0.6300	-6.49%	0.6134	-8.94%
Cardio	0.5146	0.5121	-0.48%	0.5733	11.41%
Glass	0.8538	0.8641	1.20%	0.8638	1.17%
Heart	0.5467	0.5651	3.35%	0.5267	-3.66%
InternetAds	0.6005	0.6018	0.22%	0.6221	3.60%
PageBlocks	0.6825	0.7192	5.38%	0.9313	36.45%
Pima	0.6311	0.6487	2.79%	0.6556	3.89%
SpamBase	0.4610	0.4777	3.62%	0.4982	8.06%
Stamps	0.6227	0.6863	10.22%	0.6409	2.93%
WBC	0.7873	0.7593	-3.55%	0.7910	0.47%
arrhythmia	0.7264	0.7416	2.09%	0.7420	2.15%
ionosphere	0.9065	0.9083	0.20%	0.8976	-0.98%
mammog.	0.7095	0.7003	-1.30%	0.8079	13.86%
mnist	0.6451	0.6591	2.17%	0.8565	32.77%
pima	0.5945	0.5946	0.02%	0.6125	3.02%
satellite	0.5289	0.5524	4.43%	0.6114	15.60%
satimage-2	0.6186	0.5235	-15.37%	0.9899	60.02%
speech	0.5081	0.4753	-6.45%	0.4545	-10.55%
thyroid	0.6076	0.6800	11.91%	0.9729	60.11%
vowels	0.9394	0.9515	1.29%	0.9354	-0.42%
wbc	0.9553	0.9561	0.08%	0.9526	-0.29%
Avg. $\|\Delta\|$	N/A	3.93%± 4.17%		13.35%± 18.39%	

(b) LOF (density-based)

Dataset	Ground truth	Ours	ΔOurs	Direct	ΔDirect
Annthyroid	0.5622	0.5864	4.30%	0.5718	1.71%
Cardio	0.7440	0.7496	0.76%	0.7270	-2.28%
Glass	0.6195	0.6450	4.11%	0.6515	5.16%
Heart	0.5881	0.5727	-2.62%	0.6253	6.33%
InternetAds	0.6146	0.6144	-0.04%	0.6017	-2.11%
PageBlocks	0.9047	0.8982	-0.71%	0.7949	-12.13%
Pima	0.6698	0.6339	-5.35%	0.6364	-4.98%
SpamBase	0.5506	0.5444	-1.12%	0.4572	-16.95%
Stamps	0.8989	0.8976	-0.15%	0.9217	2.53%
WBC	0.9826	0.9803	-0.24%	0.9836	0.10%
arrhythmia	0.7749	0.7806	0.73%	0.7699	-0.65%
ionosphere	0.7963	0.8339	-0.03%	0.7822	-1.78%
mammog.	0.8835	0.8711	-1.40%	0.8740	-1.08%
mnist	0.8501	0.8501	0.00%	0.7336	-13.70%
pima	0.6585	0.6113	-7.17%	0.6113	-7.17%
satellite	0.6019	0.6312	4.87%	0.6207	3.13%
satimage-2	0.9771	0.9776	0.05%	0.9078	-7.10%
speech	0.4692	0.4692	0.00%	0.5729	22.10%
thyroid	0.9455	0.9305	-1.59%	0.9452	-0.04%
vowels	0.6103	0.6072	-0.50%	0.2494	-59.14%
wbc	0.9352	0.8665	-7.35%	0.9406	0.58%
Avg. $\|\Delta\|$	N/A	2.05% ± 2.44%		8.13%± 13.13%	

(c) PCA (linear)

Dataset	Ground truth	Ours	ΔOurs	Direct	ΔDirect
Annthyroid	0.5815	0.6020	3.53%	0.5569	-4.23%
Cardio	0.5653	0.5764	1.97%	0.5839	3.29%
Glass	0.8885	0.8615	-3.03%	0.9064	2.02%
Heart	0.6003	0.5826	-2.95%	0.6220	3.62%
InternetAds	0.6297	0.6292	-0.09%	0.6329	0.51%
PageBlocks	0.8799	0.8875	0.86%	0.8834	0.39%
Pima	0.6376	0.6276	-1.56%	0.6344	-0.50%
SpamBase	0.5496	0.5535	0.70%	0.5493	-0.06%
Stamps	0.6419	0.6879	7.17%	0.7687	19.76%
WBC	0.9562	0.9674	1.17%	0.9655	0.98%
arrhythmia	0.7334	0.7668	4.55%	0.7358	0.32%
ionosphere	0.9237	0.8553	1.12%	0.9110	-1.38%
mammog.	0.7922	0.7774	-1.86%	0.7966	0.56%
mnist	0.8355	0.8460	1.26%	0.8345	-0.12%
pima	0.6245	0.6222	-0.37%	0.6200	-0.71%
satellite	0.7354	0.7186	-2.28%	0.7313	-0.55%
satimage-2	0.9980	0.9921	-0.59%	0.9905	-0.75%
speech	0.4666	0.4652	-0.30%	0.4681	0.33%
thyroid	0.9262	0.9542	3.02%	0.9283	0.22%
vowels	0.8771	0.8558	-2.43%	0.9059	3.28%
wbc	0.9383	0.9352	-0.34%	0.9826	4.72%
Avg. $\|\Delta\|$	N/A	1.96% ± 1.70%		2.30%± 2.46%	

(d) CBLOF (clustering-based)

Dataset	Ground truth	Ours	ΔOurs	Direct	ΔDirect
Annthyroid	0.6178	0.6620	7.15%	0.5961	-3.51%
Cardio	0.6794	0.6987	2.83%	0.8647	27.27%
Glass	0.7381	0.6667	-9.68%	0.8435	14.29%
Heart	0.5281	0.5988	13.39%	0.7394	40.00%
InternetAds	0.7076	0.7468	5.55%	0.7960	12.50%
PageBlocks	0.9041	0.9017	-0.26%	0.9589	6.06%
Pima	0.7398	0.7388	-0.12%	0.8190	10.71%
SpamBase	0.6419	0.5934	-7.55%	0.7530	17.31%
Stamps	0.9016	0.9175	1.76%	1.1592	28.57%
WBC	1.0000	1.0000	0.00%	1.0309	3.09%
arrhythmia	0.8439	0.8112	-3.87%	1.1685	38.46%
ionosphere	0.8264	0.7965	-3.62%	0.7128	-13.75%
mammog.	0.8636	0.8588	-0.56%	0.8750	1.32%
mnist	0.7809	0.8065	3.28%	0.8752	12.07%
pima	0.7185	0.7423	3.32%	0.8230	14.55%
satellite	0.6996	0.6824	-2.46%	0.9740	39.22%
satimage-2	0.9947	0.9960	0.13%	1.5789	58.73%
speech	0.5984	0.5385	-10.01%	0.6250	4.44%
thyroid	0.9906	0.9788	-1.20%	1.1697	18.07%
vowels	0.7810	0.6438	-17.56%	0.9001	15.25%
wbc	0.9095	0.9500	4.45%	1.0376	14.08%
Avg. $\|\Delta\|$	N/A	4.70%± 4.72%		18.73%± 14.87%	

(e) iForest (tree-based)

Dataset	Ground truth	Ours	ΔOurs	Transf.	ΔTransf.
Annthyroid	0.6616	0.6188	6.47%	0.5981	9.59%
Cardio	0.4833	0.4923	1.86%	0.5098	5.48%
Glass	0.7619	0.7381	3.12%	0.7857	3.13%
HeartDisease	0.5628	0.5411	3.86%	0.5426	3.59%
InternetAds	0.6868	0.6770	1.43%	0.6779	1.30%
PageBlocks	0.9128	0.9164	0.39%	0.8955	1.90%
Pima	0.7734	0.7831	1.25%	0.7563	2.21%
SpamBase	0.5897	0.5938	0.70%	0.5994	1.64%
Stamps	0.8857	0.9111	2.87%	0.9079	2.51%
WBC	0.9773	0.9773	0.00%	1.0000	2.32%
arrhythmia	0.8017	0.8006	0.14%	0.8323	3.81%
ionosphere	0.9294	0.9145	1.60%	0.9032	2.82%
mammog.	0.8505	0.8807	3.55%	0.8761	3.01%
mnist	0.8462	0.8544	0.97%	0.8078	4.53%
pima	0.7372	0.7497	1.70%	0.7337	0.47%
satellite	0.6879	0.7167	4.19%	0.6894	0.21%
satimage-2	0.9826	0.9966	1.42%	0.9612	2.18%
speech	0.5750	0.5678	1.25%	0.5719	0.53%
thyroid	0.9720	0.9677	0.44%	0.9632	0.90%
vowels	0.9731	0.9735	0.04%	0.9428	3.12%
wbc	0.9310	0.9333	0.25%	0.9571	2.81%
Avg. $\|\Delta\|$	N/A	1.79%± 1.67%		2.77%± 2.00%	

(f) ablation kNN (MLP vs. transformers)