Preliminary work accepted at *AAAI 2020 Workshop on Artificial Intelligence for Cyber Security (AICS)*; under revision for *KDD 2020 (ADS track)*

SUOD: Toward Scalable Unsupervised Outlier Detection

Yue Zhao,[1] **Xueying Ding,**[1] **Jianing Yang,**[2] **Haoping Bai**[2]
[1]H. John Heinz III College, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[2]Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA
zhaoy@cmu.edu, xding2@andrew.cmu.edu, jianing3@cs.cmu.edu, haopingb@andrew.cmu.edu

Abstract

Outlier detection is a key field of machine learning for identifying abnormal data objects. Due to the high expense of acquiring ground truth, unsupervised models are often chosen in practice. To compensate for the unstable nature of unsupervised algorithms, practitioners from high-stakes fields like finance, health, and security, prefer to build a large number of models for further combination and analysis. However, this poses scalability challenges in high-dimensional large datasets. In this study, we propose a three-module acceleration framework called `SUOD` to expedite the training and prediction with a large number of unsupervised detection models. `SUOD`'s Random Projection module can generate lower subspaces for high-dimensional datasets while reserving their distance relationship. Balanced Parallel Scheduling module can forecast the training and prediction cost of models with high confidence—so the task scheduler could assign nearly equal amount of taskload among workers for efficient parallelization. `SUOD` also comes with a Pseudo-supervised Approximation module, which can approximate fitted unsupervised models by lower time complexity supervised regressors for fast prediction on unseen data. It may be considered as an unsupervised model knowledge distillation process. Notably, all three modules are independent with great flexibility to "mix and match"; a combination of modules can be chosen based on use cases. Extensive experiments on more than 30 benchmark datasets have shown the efficacy of `SUOD`, and a comprehensive future development plan is also presented.

Introduction

Anomaly detection aims to identify the samples that are deviant from the general data distribution (2019). Most of the existing outlier detection algorithms are unsupervised due to the high cost of acquiring ground truth (2019). It is noted that outlier detection can be viewed as a binary classification problem under extreme imbalance (i.e., the number of outliers is way smaller than the number of normal samples). As a result, using a single unsupervised model is risky by nature; it is sensible to use a large group of unsupervised models with variations, e.g., different algorithms, varying parameters, distinct views of the datasets, etc, and more reliable results may be achieved. Outlier ensemble methods that select and combine base detectors are designed for this purpose (2013; 2014; 2017). The simplest way is to take the average or maximum values across all the detectors as the final result (2017). More complex combination can also be done in both unsupervised (2019) and semi-supervised manners (2018).

However, both training and predicting with a large number of unsupervised detectors are computationally expensive. This problem is more severe on high-dimensional large samples, especially for proximity-based algorithms that assume outliers behave very different in specific regions (2016). Most algorithms in this category, including k nearest neighbors (kNN) (2000), local outlier factor (LOF) (2000), and local outlier probabilities (LoOP) (2009), operate in Euclidean space and suffer from the curse of dimensionality. They can be prohibitively slow or even fail to work completely. The effort has been made to project high-dimensional data to lower subspaces (2001), like simple Principal Component Analysis (PCA) (2003) and more complex subspace method HiCS (2012). Engineering cures have been explored as well—the process can be expedited by parallelization on multiple workers (e.g., CPU cores) (2005; 2014). Recently, knowledge distillation emerges as a popular way of compressing large neural network models (2015), while its usage in outlier detection is still limited.

The aforementioned treatments face various challenges. First, deterministic projection methods, e.g., PCA, are fast but not ideal for building a large number of diversified outlier detectors—it results in the same subspace that cannot induce diversity for outlier ensembles. Complex projection and subspace methods may bring performance improvement for outlier mining, but the generalization capacity is often reduced with strong assumptions. They are not suited for general-purpose outlier detector acceleration. Existing parallelization learning frameworks can be inefficient if training and prediction task assignments are not balanced among workers. In fact, a group of heterogeneous models can have significantly different computational cost. As a simple example, let us split 100 heterogeneous models into 4 groups (cores) for training. If group #2 takes significantly longer time than the others to finish, it will behave like the bottleneck of the system. More formally, the imbalanced task

scheduling amplify the impact of slower worker(s) in a system; the system efficiency is curbed by the slowest group. As we have shown in the later section, the existing parallel task scheduling algorithms in popular machine learning libraries like scikit-learn (2011) may be inefficient under this setting. In addition to these limitations, unsupervised models such like LOF can be slow (high time complexity), or even inappropriate, to predict on unseen data samples by nature. Another downside of unsupervised and non-parametric models is their limited interpretability. In summary, training and predicting with a large number of heterogeneous unsupervised models is computationally expensive, inefficient in parallelization, and limited in interpretability.

To tap the gap, we propose a three-module acceleration framework called `SUOD` that leverages random projection, pseudo-supervised approximation, and balanced parallel scheduling for scalability. For high-dimensional data, `SUOD` generates a random low-dimensional subspace for each unsupervised model by Johnson-Lindenstrauss projection, on which the model is then trained. To improve the training and prediction efficiency in distributed environment, a balanced parallel scheduling mechanism is designed. The key idea is to forecast the running time of each model so that the workload could be nearly even distributed among workers. If prediction is needed for unseen data, lower cost supervised regressors are initialized to approximate complex unsupervised models—the supervised models are trained on the original feature space using the outlier scores generated by unsupervised models as the "pseudo ground truth". The rationale behind is efficient supervised models are faster for prediction and usually more interpretable; it can be viewed as a way of distilling knowledge from unsupervised models (2015). Notably, all these three modules are designed to be fully independent for model acceleration from complementary perspectives. They have great flexibility to be mixed for different needs.

In this work, we make the following contributions:

1. Examine the effect of various deterministic and random projection methods on varying size and dimension datasets, and identify the applicable cases of using them for faster execution and diversity induction.
2. Identify an imbalanced scheduling issue in existing distributed learning systems for heterogeneous detectors and fix it by a new scheduling schema.
3. Conduct extensive experiments to analyze the results of using pseudo-supervised regression models for approximating unsupervised outlier detectors. To our best knowledge, this is the first research attempt in outlier detection setting.
4. Demonstrate the effectiveness of the three modules independently, and the extensibility of combining them together as a scalable training and prediction framework.
5. To foster reproducibility, all code, figure, and datasets used in this study are openly shared[1]. A scalable implementation will also be included in PyOD (2019) soon.

[1]https://github.com/yzhao062/SUOD

Related Works

Outlier Detection and Outlier Ensembles

Anomaly detection has numerous important applications in various fields, such as rare disease detection (2018), fraudulent online review analysis (2013), and network intrusion detection (2003). Despite, detecting outliers is a challenging classification task due to multiple reasons (2019). First, anomalies only consist of a small portion of the entire data– extreme data imbalance incurs difficulty. Second, the limited amount of data and available labels impede learning data representation accurately. Third, the definition of outliers can be ambiguous; outliers may be heterogeneous that should be treated as a multi-class problem.

Most of the existing detection algorithms are unsupervised as ground truth is often absent; acquiring labels can be prohibitively expensive in practice. As a result, there are a few established unsupervised anomaly detection algorithms like Isolation Forest (2008), Local Outlier Factor (LOF) (2000), and Angle-based Outlier Detection (ABOD) (2009), with different assumptions of the underlying data. Regarding unsupervised deep models like autoencoders and generative adversarial networks (2019), the amount of accessible data limits their effectiveness on learning representations. No algorithm could always outperform as the assumptions may be incorrect, and it is hard to asses without ground truth.

Therefore, relying on a single unsupervised model has an inherently high risk and outlier ensembles that leverage a group of detectors become increasingly popular (2013; 2014). There are a group of unsupervised outlier ensemble frameworks proposed in the last several years from simple average, maximization, weighted average, second-phase combination methods (2017) to more complex selective models like SELECT (2016) and LSCP (2019). Although unsupervised outlier ensembles methods can be effective in certain cases, they could not incorporate the existing ground truth information regardless of its richness. As a result, a group of semi-supervised detection frameworks that leverage existing labels and enhance the data representation by unsupervised models are proposed. The representative ones include BORE (2014) and XGBOD (2018). They use unsupervised outlier detection scores as additional features to enhance the original feature space, which can be thought as unsupervised feature engineering or representation learning (extraction). It is noted that for both unsupervised and supervised outlier ensembles, a large group diversified unsupervised base detectors are needed—`SUOD` is therefore designed to facilitate this process.

Knowledge Distillation and Model Approximation

Knowledge distillation refers to the notion of compressing a or an ensemble of large, often cumbersome model(s) into a small and more interpretable model. This is often done by training an ensemble of large models (can be seen as "Teachers") on a large dataset, followed by using a small and simple model ("Student") to learn the output of the ensemble. There are two main motivations behind knowledge distillation: (i) to reduce deployment-time computational cost by replacing the large models with a small model and (ii) to in-

crease interpretability as simple models are often more easy to be understood by human. This strategy has seen success in tasks including computer vision (2014), automatic speech recognition (2015), and neural machine translation (2016). It is noted that the proposed SUOD framework shares a similar concept as knowledge distillation for computational cost optimization, but comes with a few fundamental differences as described in Algorithm Design section.

Algorithm Design

The proposed SUOD contains three independent modules. As shown in Algorithm 1, the modules may be enabled if specific conditions are met. For high-dimensional data, SUOD randomly project the original input onto lower-dimensional spaces (***Module I***). For expediting the training and prediction with a large number of models, a balanced parallel scheduling mechanism is proposed (***Module II***). If prediction on new samples is needed, a efficient supervised regressor may be initialized to approximate the output of each costly unsupervised detector for prediction (***Module III***).

Module I: Random Projection

A widely used algorithm to alleviate the curse of dimensionality on high-dimensional data is the Johnson-Lindenstraus (JL) projection (1984), although its use in outlier mining is still unexplored. JL projection is a simple compression scheme without heavy distortion on the Euclidean distances of the data. Its built-in randomness is also useful for inducing diversity for outlier ensembles. Despite, projection may be less useful or even detrimental for methods like Isolation Forests and HBOS that rely on subspace splitting.

This linear transformation is defined as: given a set of data $X = \{x_1, x_2, ...x_n\}$, each $x_i \in \mathbb{R}^d$, let W be a $k \times d$ matrix with each entry drawing independently from a $\mathcal{N}(0, 1)$ distribution or a Rademacher distribution. Then the JL projection is a function $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$ such that $f(x_i) = \frac{1}{\sqrt{k}} W x_i$. JL projection randomly projects high-dimensional data to lower dimension subspaces, but preserve the distance relationship between points. In fact, if we fix some $v \in \mathbb{R}^d$, and let W be the $k \times d$ matrix such that each entry is from $\mathcal{N}(0, 1)$. For every $\epsilon \in (0, 3)$, we have:

$$P\left[(1-\epsilon)\|v\|^2 \leq \|\frac{1}{\sqrt{k}} W v\|^2 \leq (1+\epsilon)\|v\|^2\right] \leq 2e^{-\epsilon^2 \frac{k}{6}} \tag{1}$$

Furthermore, fix v to be the differences between vectors. Then, the above bound also shows that for a finite set of N vectors $X = \{x_1, x_2, ...x_n\} \subseteq \mathbb{R}^d$, the pairwise Euclidean distance is preserved within a factor of $(1 \pm \epsilon)$, if we reduce the vectors to $k = \mathcal{O}(\frac{log(N)}{\epsilon^2})$ dimensions.

Four JL projection methods are therefore introduced for their great property in compression and diversity induction: (i) *basic*: the transformation matrix is generated by standard Gaussian; (ii) *discrete*: the transformation matrix is picked randomly from Rademacher distribution (uniform in $\{-1, 1\}$); (iii) *circulant*: the transformation matrix is obtained by rotating the subsequent rows from the first row which is generated from standard Gaussian and (iv) *toeplitz*: the first row and column of the transformation matrix are generated from standard Gaussian, and each diagonal uses a constant value from the first row and column.

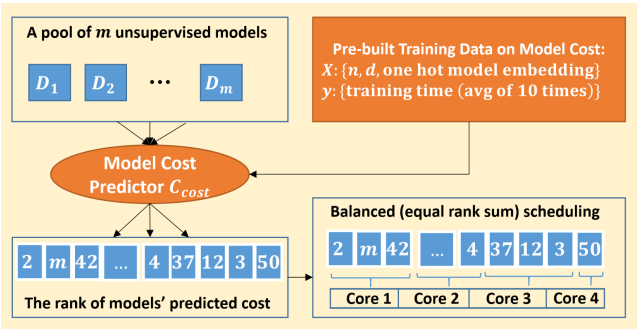
Let $X \in \mathcal{R}^{n \times d}$ denote a dataset with n points and d features. In this work, we only invoke the projection if the data dimension exceeds 20 (dimension threshold $\theta = 20$) and reduce the original dimension by half (projection dimension $k = \frac{1}{2}d$). SUOD check whether d is higher the projection threshold θ. If so, a JL transformation matrix $W \in \mathcal{R}^{k \times d}$ is initialized by one of the JL projection methods and X is projected onto the k dimension subspace by W.

Module II: Balanced Parallel Scheduling

Balanced Parallel Scheduling (BPS) aims for assigning training and prediction tasks more evenly based on the model costs, across all available workers. For instance, one may train 25 detectors with varying parameters from each of the four algorithm groups {*k*NN, LOF, ABOD, OCSVM}, resulting in 100 models in total. The existing parallel frameworks, e.g., the voting machine in scikit-learn (2011), will simply split the the models into 4 subgroups by order and schedule the first 25 models (all *k*NNs) on Core 1 (worker 1), the next 25 models on Core 2, etc. This does not account for the fact that within a group of heterogeneous detectors, the computational cost varies. Scheduling the task with equal number of models can result in highly imbalanced tasks. In the worst case scenario, one worker may be significantly slower than the rest, and the entire process halts. Obviously, this problem applies to both training and prediction stage.

The proposed BPS heuristic focuses on designing a more balanced schedule among workers. Ideally, all workers can finish the scheduled tasks within the similar duration of time and return the result. As shown in Fig. 1, we build a *model cost predictor* C_{cost} to forecast the model running time (sum of 10 trails) given the input data size, input data dimension, and the algorithm embedding (one-hot). Given the time and resource limitation, we built a training set with 11 algorithms on 47 benchmark datasets (see Experiment section for details), and a random forest regressor (2001) is trained on the dataset with 10-fold cross validation. Although the fitted regressor does not have a high R^2 score, the Spearman's Rank correlation (1904) consistently shows high value ($r_s > 0.9$) with low p-value ($p < 0.0001$). This implies that even the cost predictor C_{cost} could not predict the running time precisely, it can predict the rank of the running time with high accuracy. As a result, we propose a scheduling heuristic by enforcing nearly equal sum of the rank on running time. Given there are m models to be trained, cost predictor C_{cost} is first invoked to forecast the time cost for a given model $\mathcal{D}_i$ in $\mathcal{D}$ as $C_{cost}(\mathcal{D}_i)$. After the prediction is done, the predicted time is converted to a rank in $[1, m]$. If there are t cores (workers), each worker will be assigned a group of models to achieve the objective of minimizing the workload imbalance among workers (Eq. 2). So each group has a rank sum at around $\frac{m^2+m}{2t}$. One additional advantage of using rank is transferability: the running time will vary on different machines but the relative rank should preserve.

Figure 1: Flowchart of Balanced Parallel Scheduling



$$\min_{\mathcal{W}} \sum_{i=1}^{t} \left| \sum_{D_j \in \mathcal{W}_i} C_{cost}(\mathcal{D}_i) - \sum_{l=1}^{m} C_{cost}(\mathcal{D}_l) \right| \quad (2)$$

Module III: Pseudo-Supervised Approximation

Once the unsupervised models have been fitted on the reduced feature space generated in ***Module I*** or the original space (if no random projection is involved). `SUOD` can approximate and replace each of **costly unsupervised model** by a **faster supervised regression model** for predicting on unseen samples. In other words, not all unsupervised models should be replaced but only the expensive ones. There is no efficiency incentive to approximate fast algorithms like Isolation Forest. However, most of the proximity-based algorithms like *k*NN and LOF have high time complexity for prediction (upper bounded by $\Theta(nd)$), which can be effectively replaced by fast supervised models like random forest (upper bounded by $\Theta(dp)$ where p denotes the number of base trees). This "pseudo-supervised" model uses the output of unsupervised models as "the pseudo ground truth"—the goal is to approximate and find a better mapping from the original input to "the output of an unsupervised model". Ensemble-based tree models are recommended for their outstanding scalability, robustness to overfitting, and interpretability (e.g., feature importance). Notably, this process can also be viewed as using supervised regressors to distill knowledge from unsupervised models. However, our approximation is different from the established knowledge distillation mainly in three aspects. First, our approximation works in a fully unsupervised manner unlike the classic distillation under supervised settings. Second, our "teacher" and "student" models have totally different architectures with little to no similarity. For instance, we use random forest (an ensemble tree model) to approximate LOF (a proximity-based density estimation model). Third, our approximation leads to a clear interpretability improvement, whereas the student models in neural networks lack that. See Appendix **??**.

As shown in Algorithm 1, for each trained unsupervised model $\mathcal{D}_i$, a supervised regressor $\mathcal{R}_i$ might be built with the original input X and the pseudo ground truth (the output of $\mathcal{D}_i$). The prediction on unseen data will be then made by $\mathcal{R}$. This approximation shows multiple benefits:

1. Compared with non-parametric unsupervised models, parametric supervised models may have lower space complexity and faster prediction speed.
2. Supervised models generally show better interpretability compared with unsupervised counterparts. For instance, random forest used in this work can generate feature importance automatically to facilitate understanding.
3. Not all unsupervised models are appropriate for making prediction. Taking LOF as an example, the fitted model is not supposed to predict on unseen data. A supervised approximator may be used for prediction in this case.

Algorithm 1 Scalable Unsupervised Outlier Detection

Inputs: a group of m unsupervised outlier detectors $\mathcal{D}$; d dimension training data X_{train} and test data X_{test} (optional); projection threshold θ; projection dimension k; pre-trained cost predictor C_{cost}; # of workers t

Outputs: fitted unsupervised models $\mathcal{D}$; fitted pseudo-supervised regressors $\mathcal{R}$ (optional); prediction results on test data $\hat{y}_{test}$ (optional)

for Each detector $\mathcal{D}_i$ in $\mathcal{D}$ **do**
  if $d > \theta$ **then** *// enable random projection*
    Generate random subspace ψ_i by JL projection on X_{train} (see ***Module I***)
  else $d < \theta$ *// disable random projection*
    Use the original feature space $\psi_i := X_{train}$
  end if
end for
if Parallel learning == True **then**
  Train each $\mathcal{D}_i$ on the corresponding ψ_i by Balanced Parallel Scheduling on t workers (Eq. (2) and Fig. 1)
end if
Return $\mathcal{D}$
if X_{test} is presented **and** Approximation == True **then**
  Acquire the pseudo ground truth $target^{\psi_i}$ as the output of $\mathcal{D}_i$ on ψ_i: $target^{\psi_i} := \mathcal{D}_i(\psi_i)$
  for Each detector $\mathcal{D}_i$ in $\mathcal{D}$ **do**
    Initialize a supervised regressor $\mathcal{R}_i$
    Fit $\mathcal{R}_i$ by $\{X, target^{\psi_i}\}$ in pseudo-supervised approximation described in ***Module III***
    $\hat{y}^i_{test} = \mathcal{R}_i$.predict$(X_{test})$
  end for
  Return $\hat{y}_{test}$ and fitted regressors $\mathcal{R}$
end if

Numerical Experiments and Discussion

In this preliminary study, three independent experiments are conducted to understand: (i) how will random projection methods affect the performance of outlier detection algorithms; (ii) whether the proposed parallel scheduling algorithm brings performance improvement over the existing approaches and (iii) will pseudo-supervised models lead to degraded prediction performance compared with the original unsupervised models? Because all three modules are independent and can be combined seamlessly, it is assumed that

the (partly) combined framework should also work if individual components manage to work.

Datasets, Evaluation Metrics, and Implementation

More than 30 outlier detection benchmark datasets are used in this study[1,2]); the detail is available on online supplementary due to the space limit. The data size n varies from 219 (**Glass**) to 567,479 (**HTTP**) samples and the dimension d ranges from 3 to 400. For both random projection and parallel scheduling experiments, full datasets are used for training. For the pseudo-supervised approximation experiments, 60% of the data is used for training and the remaining 40% is set aside for validation. For all experiments, performance is evaluated by taking the average of 10 independent trials using area under the receiver operating characteristic (ROC) and precision at rank n (P@N). Both metrics are widely used in outlier research (2008; 2014; 2016; 2017; 2018; 2019; 2019).

All the unsupervised models are from Python Outlier Detection Toolbox (`PyOD`), a popular library for outlier mining (2019). Supervised regressors and utility functions are from standard libraries (`scikit-learn` and `numpy`). For a fair comparison, none of the models involve parameter tuning process—the default values are used. As all three modules involve time profiling, the same machine (Intel i7-9700 @ 3.00 GHZ; 32 GB RAM) is used for a fair comparison.

Comparison among Projection Methods

To evaluate the effect of projection methods on outlier detector performance, we choose three expensive detection algorithms (ABOD, LOF, and *k*NN) to measure their execution time, ROC, and P@N with different projection methods. All three methods directly or indirectly measure sample similarity in the Euclidean space , e.g., pairwise sample distances, which is susceptible to the curse of dimensionality and projection may be particularly helpful.

Table 1 shows the comparison among four JL projection variations with original (no projection is used), PCA, and RS (randomly select k features from the original d features, used in Feature Bagging (2005) and LSCP). First, all projection methods show superiority regarding time cost. Second, using RS method comes with high instability, and shows performance decrease on all three datasets. This observation agrees with the finding by Zhao et al. (2019), in which they used an ensemble projection to overcome the instability. Third, PCA is slightly faster than JL projection methods, although the detector performance by PCA projection is not as good as the ones with JL projections as shown in subtable (b)-(i). Moreover, PCA as a deterministic method, may not be ideal for inducing diversity in outlier ensembles, as it always result in the same sets of subspaces. Fourth, among all four JL projection methods, *circulant* and *toeplitz* outperform in most cases. Since *toeplitz* is slightly faster than *circulant*, it is a reasonable choice for reducing dimensionality and inducing diversity for the models that are susceptible to the curse of dimensionality.

[1]ODDS Library: http://odds.cs.stonybrook.edu

[2]DAMI Datasets: http://www.dbs.ifi.lmu.de/research/outlier-evaluation/DAMI

The Effect of Balanced Parallel Scheduling

To verify the effectiveness of the proposed BPS algorithm, we run the following experiments by varying: (i) the size (n) and the dimension (d) of the datasets, (ii) the number of estimators to train (m) and (iii) the number of CPU cores (t). The time elapsed is measured in seconds. Due to the space limit, we only show the comparison between the simple scheduling and BPS on **Cardio** ($n = 1831, d = 21$), **PageBlocks** ($n = 5393, d = 10$), and **Pendigits** ($n = 6870, d = 16$), by setting $t \in \{2, 4, 6\}$ and $m \in \{100, 500\}$.

More results can be found on the online supplementary, and the conclusion holds for all tested datasets. Table 2 shows that the proposed BPS has a clear edge over the simple scheduling mechanism (denoted as **Simple** in the tables) that equally splits the tasks by order. It yields a significant time reduction (denoted as **% RED** in the tables), and gets more significant if more estimators and cores are involved. For instance, if 500 estimators and 6 cores are used, the time reduction over the simple scheduling is more than 40%. This agrees with our assumption that the imbalanced task scheduling will lead to more inefficient consequences with the increasing number of estimators and workers, and will therefore benefit more from the proposed BPS heuristic.

The Analysis of Pseudo-Supervised Approximation

To better understand the behavior of the pseudo-supervised approximation, we first generate 200 synthetic points with Normal distribution for outliers (40 points) and Uniform distribution for normal samples (160 points). In Fig. 2, we plot the decision surfaces of unsupervised models and their supervised approximators (random forest regressor). It is clear that the decision surfaces are different and some regularization effect appears (lower errors on Feature Bagging and *k*NN). One of the assumptions is that the approximation process improves the generalization ability of the model by "ignoring" the overfitted points. This fails to work with ABOD because it has a extremely coarse decision surfaces to approximate (see Fig. 2).

Table 3 and 4 compare prediction performance between the original unsupervised models and pseudo-supervised approximators on 8 datasets with 6 algorithms. These algorithms are known to be more computationally expensive than the supervised random forest regressors. The approximators with performance degradation are highlighted in bold and italicized in the tables. The prediction time comparison is omitted due to space limit, but the gain is clear (see online supplementary). Therefore, the focus is put on prediction ROC and P@N to see whether the approximators could predict unseen data as good as the original unsupervised models. The acceptable threshold of performance degradation between an approximator and its original unsupervised models is set as $[0, 0.01]$ and any negative difference larger than 0.01 will be regarded as degradation. The tables reveal that not all the algorithms can be approximated well by supervised regressors: ABOD has a performance decrease regarding ROC on multiple datasets. ABOD is a linear mod-

Table 1: Comparison of various projection methods on different outlier detectors and datasets

(a) ABOD on **MNIST**

Method	Time	ROC	PRN
original	12.89	0.80	0.39
PCA	8.93	0.81	0.37
RS	8.27	0.74	0.32
basic	8.94	0.80	0.38
discrete	8.86	0.80	0.39
circulant	9.33	0.80	0.38
toeplitz	8.96	0.80	0.38

(b) ABOD on **Satellite**

Method	Time	ROC	PRN
original	4.03	0.59	0.41
PCA	3.01	0.62	0.44
RS	3.53	0.63	0.44
basic	3.10	0.64	0.45
discrete	3.12	0.65	0.46
circulant	3.14	0.66	0.48
toeplitz	3.14	0.66	0.47

(c) ABOD on **Satimage-2**

Method	Time	ROC	PRN
original	3.68	0.85	0.28
PCA	2.70	0.88	0.30
RS	3.20	0.89	0.28
basic	2.78	0.91	0.29
discrete	2.79	0.91	0.31
circulant	2.85	0.91	0.29
toeplitz	2.83	0.92	0.30

(d) LOF on **MNIST**

Method	Time	ROC	PRN
original	7.64	0.68	0.29
PCA	4.92	0.67	0.27
RS	3.65	0.63	0.23
basic	4.87	0.70	0.31
discrete	5.21	0.70	0.32
circulant	5.06	0.69	0.31
toeplitz	4.97	0.71	0.31

(e) LOF on **Satellite**

Method	Time	ROC	PRN
original	0.82	0.55	0.38
PCA	0.23	0.54	0.36
RS	0.39	0.54	0.37
basic	0.31	0.54	0.37
discrete	0.32	0.54	0.37
circulant	0.39	0.55	0.37
toeplitz	0.37	0.54	0.37

(f) LOF on **Satimage-2**

Method	Time	ROC	PRN
original	0.79	0.54	0.07
PCA	0.20	0.52	0.04
RS	0.37	0.53	0.08
basic	0.29	0.52	0.08
discrete	0.30	0.53	0.07
circulant	0.43	0.59	0.11
toeplitz	0.32	0.54	0.09

(g) *k*NN on **MNIST**

Method	Time	ROC	PRN
original	7.13	0.84	0.42
PCA	3.92	0.84	0.40
RS	3.33	0.77	0.34
basic	4.17	0.84	0.42
discrete	4.11	0.84	0.41
circulant	4.13	0.84	0.41
toeplitz	4.11	0.84	0.42

(h) *k*NN on **Satellite**

Method	Time	ROC	PRN
original	0.71	0.67	0.49
PCA	0.18	0.67	0.50
RS	0.31	0.68	0.49
basic	0.24	0.68	0.49
discrete	0.25	0.69	0.50
circulant	0.33	0.70	0.50
toeplitz	0.30	0.70	0.51

(i) *k*NN on **Satimage-2**

Method	Time	ROC	PRN
original	0.68	0.94	0.39
PCA	0.15	0.94	0.39
RS	0.29	0.94	0.38
basic	0.23	0.94	0.38
discrete	0.20	0.95	0.37
circulant	0.36	0.96	0.37
toeplitz	0.25	0.96	0.39

els that look for a lower-dimensional subspace to embed the normal samples (2016), so the approximation may not work if it has extremely complex decision surfaces as mentioned before. In contrast, proximity-based models that aim to identify specific Euclidean regions in which outliers are different, benefit from the approximation. Both table shows, *k*NN, LoF, and A*k*NN (average *k*NN) experience an performance gain. Specifically, all three algorithms yield around 100% ROC increase on **HTTP**. Other algorithms, such as Feature Bagging and CBLOF, the ROC and PRC performances stay within the acceptable range. In other words, it is useful to perform pseudo-supervised approximation for these estimators as the time efficiency is improved at little to no loss in accuracy. Through the visualization and quantitative comparisons, we believe that the proposed pseudo-supervised approximation is meaningful for prediction acceleration.

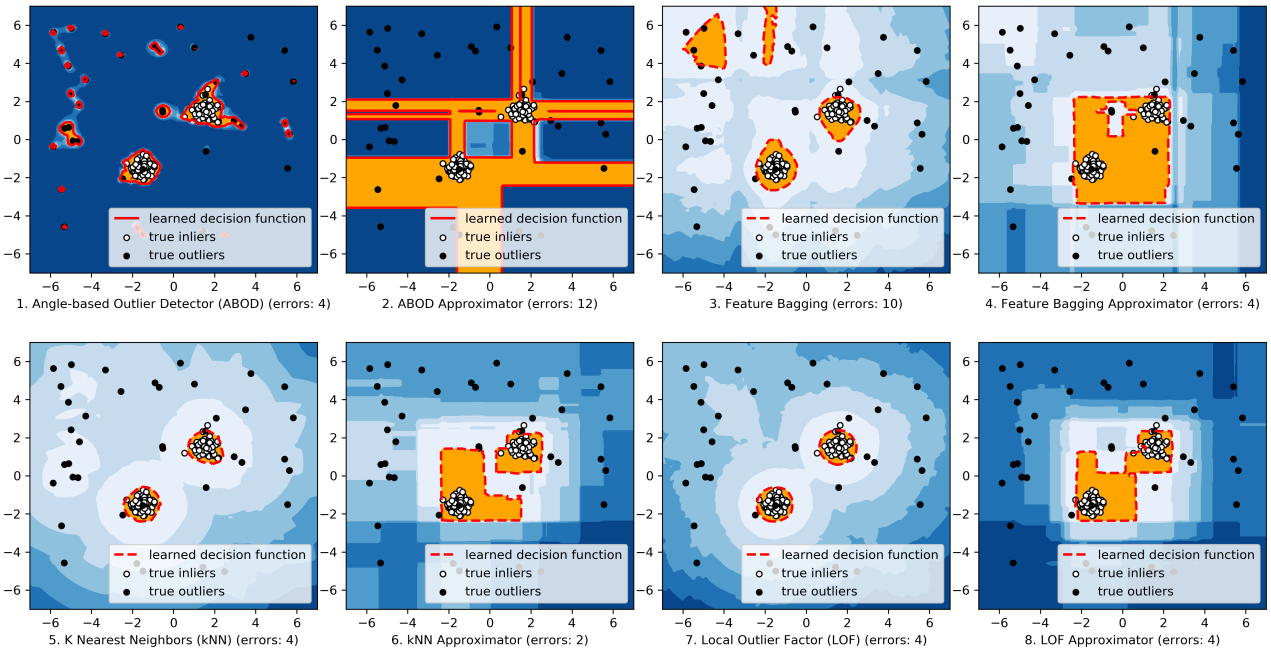
Conclusion and Future Directions

In this work, a three-module framework called `SUOD` is proposed to accelerate the training and prediction with a large number of unsupervised anomaly detectors. The three modules in `SUOD` focus on different perspectives of scalability enhancement: (i) Random Projection module gener-

Table 2: Comparison between simple scheduling and BPS

n	d	m	t	**Simple**	**BPS**	**% RED**
1831	21	100	2	26.33	19.85	24.61
1831	21	100	4	17.93	13.69	23.65
1831	21	100	6	19.16	15.23	20.51
1831	21	500	2	100.51	72.16	28.21
1831	21	500	4	80.38	39.46	50.91
1831	21	500	6	55.3	32.78	40.72
5393	10	100	2	51.11	35.17	31.19
5393	10	100	4	42.49	16.23	61.80
5393	10	100	6	38.45	16.97	55.86
5393	10	500	2	197.84	137.46	30.52
5393	10	500	4	167.36	76.14	54.51
5393	10	500	6	127.08	66.29	47.84
6870	16	100	2	80.89	67.23	16.89
6870	16	100	4	68.31	35.29	48.34
6870	16	100	6	49.19	24.18	50.84
6870	16	500	2	336.54	286.14	14.98
6870	16	500	4	345.69	164.02	52.55
6870	16	500	6	211.34	113.13	46.47

Figure 2: Comparison among unsupervised models and their pseudo-supervised counterparts



ates lower-dimensional subspaces to alleviate the curse of dimensionality using toeplitz Johnson-Lindenstraus projection; (ii) Balanced Parallel Scheduling module ensures that nearly equal amount of workloads are assigned to multiple workers in parallel training and prediction and (iii) Pseudo-supervised Approximation module could accelerate costly unsupervised models' prediction speed by replacing them by scalable supervised regressors, which also brings the extra benefit regarding interpretability and storage cost. The extensive experiments on more than 30 benchmark datasets empirically show the great potential of SUOD, and many intriguing results are observed. To improve the model reproducibility and accessibility, all code, figures, implementation for demo and production, will be released[1].

Many investigations are underway. First, we would like to demonstrate SUOD's effectiveness as an end-to-end framework by combining three modules, and provide an easy to use toolkit in Python. Three additional experiments may be conducted to show that SUOD is useful for: (i) simple combination like majority vote and maximization (2020); (ii) more complex unsupervised model combination like LSCP (2019) and (iii) supervised outlier combination algorithms such as XGBOD (2018). Second, although we provide a pre-built *model cost predictor* as part of the framework, a re-trainable cost predictor is expected so practitioners can make accurate prediction on their machines. Third, we would further emphasize the interpretability provided by the pseudo-supervised approximation, which can be beyond simple feature importance provided in tree regressors. Fourth, we see there is room to investigate why and how the pseudo-supervised approximation could work in a more strict and theoretical way. Specifically, we want to know how to choose supervised regressors and under what conditions the approximation could work. This study, as the first step, empirically shows that proximity-based models benefit from the approximation, whereas linear models may not. Lastly, we want to verify that SUOD can work with real-world use cases. We are in contact with a pharmaceutical consultancy firm to access their rare disease detection datasets (which can be viewed as an outlier detection task). Microsoft Malware Dataset[2] is also chosen to explore SUOD's applicability.

[1]https://github.com/yzhao062/SUOD

[2]https://www.kaggle.com/c/microsoft-malware-prediction

References

Achlioptas, D. 2001. Database-friendly random projections. In *PODS*, 274–281. ACM.

Aggarwal, C. C., and Sathe, S. 2017. *Outlier ensembles: An introduction*. Springer, 1st edition.

Aggarwal, C. C. 2013. Outlier ensembles: position paper. *ACM SIGKDD Explorations Newsletter* 14(2):49–58.

Aggarwal, C. C. 2016. *Outlier Analysis*. Springer.

Akoglu, L.; Chandy, R.; and Faloutsos, C. 2013. Opinion fraud detection in online reviews by network effects. In *ICWSM*.

Breiman, L. 2001. Random forests. *Machine learning* 45(1):5–32.

Table 3: Test ROC scores of unsupervised models and their pseudo-supervised approximators (avg of 10 independent trials)

Dataset	**Annthyroid**		**Breastw**		**Cardio**		**HTTP**		**MNIST**		**Pendigits**		**Pima**		**Satellite**	
Model	Orig	Appr	Orig	Appr	Orig	Appr	Orig	Appr	Orig	Appr	Orig	Appr	Orig	Appr	Orig	Appr
ABOD	0.83	***0.71***	0.92	0.93	0.63	***0.53***	0.15	***0.13***	0.81	***0.79***	0.67	0.82	0.66	0.70	0.59	0.68
CBLOF	0.67	0.68	0.96	0.98	0.73	0.76	1.00	1.00	0.85	0.89	0.93	0.93	0.63	0.68	0.72	0.77
FB	0.81	***0.45***	0.34	***0.10***	0.61	0.70	0.34	0.97	0.72	0.83	0.39	0.51	0.59	0.63	0.53	0.64
KNN	0.80	0.79	0.97	0.97	0.73	0.75	0.19	0.85	0.85	0.86	0.74	0.87	0.69	0.71	0.68	0.75
AKNN	0.81	0.82	0.97	0.97	0.67	0.72	0.19	0.88	0.84	0.85	0.72	0.87	0.69	0.71	0.66	0.74
LOF	0.74	0.85	0.44	0.45	0.60	0.68	0.35	0.75	0.72	0.76	0.38	0.47	0.59	0.65	0.53	0.66

Table 4: Test P@N scores of unsupervised models and their pseudo-supervised approximators (avg of 10 independent trials)

Dataset	**Annthyroid**		**Breastw**		**Cardio**		**HTTP**		**MNIST**		**Pendigits**		**Pima**		**Satellite**	
Model	Orig	Appr	Orig	Appr	Orig	Appr	Orig	Appr	Orig	Appr	Orig	Appr	Orig	Appr	Orig	Appr
ABOD	0.31	***0.08***	0.80	0.83	0.27	***0.20***	0.00	0.00	0.40	0.27	0.05	0.05	0.48	0.52	0.41	0.46
CBLOF	0.25	**0.24**	0.86	0.90	0.31	0.34	0.02	**0.01**	0.42	0.48	0.35	0.36	0.43	0.48	0.54	0.57
FB	0.24	***0.02***	0.03	0.07	0.23	0.26	0.02	0.04	0.34	0.36	0.03	0.07	0.37	0.44	0.37	0.42
KNN	0.30	0.32	0.89	0.89	0.37	0.46	0.03	0.03	0.42	0.45	0.08	***0.06***	0.47	0.47	0.49	0.53
AKNN	0.30	0.33	0.88	0.89	0.34	0.40	0.03	0.03	0.41	0.45	0.05	0.13	0.48	0.49	0.47	0.52
LOF	0.27	0.36	0.19	0.35	0.23	0.23	0.01	0.03	0.33	0.32	0.03	0.08	0.40	0.44	0.37	0.42

Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; and Sander, J. ö. r. 2000. LOF: Identifying Density-Based Local Outliers. *SIGMOD* 1–12.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Johnson, W. B., and Lindenstrauss, J. 1984. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics* 26(189-206):1.

Keller, F.; Muller, E.; and Bohm, K. 2012. Hics: High contrast subspaces for density-based outlier ranking. In *ICDE*.

Kim, Y., and Rush, A. M. 2016. Sequence-level knowledge distillation. *EMNLP*.

Kriegel, H.-P.; Kr ö ger, P.; Schubert, E.; and Zimek, A. 2009. LoOP: local outlier probabilities. *CIKM* 1649–1652.

Lazarevic, A., and Kumar, V. 2005. Feature bagging for outlier detection. In *KDD*, 157–166. ACM.

Lazarevic, A.; Ertoz, L.; Kumar, V.; Ozgur, A.; and Srivastava, J. 2003. A comparative study of anomaly detection schemes in network intrusion detection. In *SDM*, 25–36.

Li, W.; Wang, Y.; Cai, Y.; Arnold, C.; Zhao, E.; and Yuan, Y. 2018. Semi-supervised rare disease detection using generative adversarial network. In *NeurIPS Workshop*.

Liu, Y.; Li, Z.; Zhou, C.; Jiang, Y.; Sun, J.; Wang, M.; and He, X. 2019. Generative adversarial active learning for unsupervised outlier detection. *TKDE*.

Liu, F. T.; Ting, K. M.; and Zhou, Z.-H. 2008. Isolation forest. In *ICDM*, 413–422. IEEE.

Lozano, E., and Acufia, E. 2005. Parallel algorithms for distance-based and density-based outliers. In *ICDM*.

Micenková, B.; McWilliams, B.; and Assent, I. 2014. Learning outlier ensembles: The best of both worlds–supervised and unsupervised. In *SIGKDD Workshop*, 51–54.

Oku, J.; Tamura, K.; and Kitakami, H. 2014. Parallel processing for distance-based outlier detection on a multi-core cpu. In *2014 IEEE 7th International Workshop on Computational Intelligence and Applications (IWCIA)*, 65–70. IEEE.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. 2011. Scikit-learn: Machine learning in python. *JMLR* 12(Oct):2825–2830.

Ramaswamy, S.; Rastogi, R.; Shim, K.; Hill, M.; Shim, K.; and Ramaswamy, Sridhar, Rajeev rastogi, K. S. 2000. Efficient algorithms for mining outliers from large data sets. *ACM SIGMOD Record* 29(2):427–438.

Rayana, S., and Akoglu, L. 2016. Less is more: Building selective anomaly ensembles. *TKDD* 10(4):42.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

Shyu, M.-L.; Chen, S.-C.; Sarinnapakorn, K.; and Chang, L. 2003. A novel anomaly detection scheme based on principal component classifier. Technical report.

Spearman, C. 1904. The proof and measurement of association between two things. *AJP*.

Zhao, Y., and Hryniewicki, M. K. 2018. Xgbod: Improving supervised outlier detection with unsupervised representation learning. In *IJCNN*. IEEE.

Zhao, Y.; Nasrullah, Z.; Hryniewicki, M. K.; and Li, Z. 2019. LSCP: locally selective combination in parallel outlier ensembles. In *SDM*, 585–593.

Zhao, Y.; Wang, X.; Cheng, C.; and Ding, X. 2020. Combining machine learning models and scores using combo library. In *AAAI*.

Zhao, Y.; Nasrullah, Z.; and Li, Z. 2019. PyOD: A python toolbox for scalable outlier detection. *JMLR* 20(96):1–7.

Zimek, A.; Campello, R. J.; and Sander, J. 2014. Ensembles for unsupervised outlier detection: challenges and research questions a position paper. *ACM SIGKDD Explorations Newsletter* 15(1):11–22.