ActiveAD: Enhancing Anomaly Detection in Tabular Data through Active Learning Strategies

Haoyan Luo[1,2] , **Xiaofan Gui**[2] , **Wei Cao**[2] , **Jiang Bian**[2]
[1]School of Data Science, The Chinese University of Hong Kong, Shenzhen
[2]Microsoft Research Asia
haoyanluo@link.cuhk.edu.cn, {xiaofangui, weicao, jiang.bian}@microsoft.com

Abstract

Detecting anomalies in tabular data is critical in many fields, including cybersecurity, finance, and healthcare. However, labeling data for anomaly detection is often labor-intensive and costly. *Active learning* (AL) emerges as a promising approach to mitigate these challenges, aiming to reduce the labeling cost while maintaining high detection performance. In our project, we propose a pipeline, ActiveAD, for active anomaly detection that combines various anomaly detection models and active learning querying strategies to improve the efficiency and effectiveness of identifying anomalies with limited labeled data. Benefiting from the recent study on anomaly detection benchmark, we also offer a comprehensive comparison of different active learning method performance on diverse datasets. Extensive experiments reveal the strengths and weaknesses of each method and the impact of outliers, providing valuable insights into their suitability under different conditions.

1 Introduction

Tabular anomaly detection (TAD), is an important research topic in the field of machine learning that has been applied to various domains, including finance [Ahmed *et al.*, 2016], security [Weller-Fahy *et al.*, 2014], energy [Himeur *et al.*, 2021], traffic [Djenouri *et al.*, 2018], and machine failure [Riazi *et al.*, 2019]. The primary goal of anomaly detection tasks is to identify unusual patterns or behaviors within datasets, which often represent rare events, errors, or malicious activities. The success of an anomaly detection model relies heavily on the quality and quantity of labeled data used for training.

However, acquiring labeled data can be labor-intensive, time-consuming, and expensive, especially in domains where expert knowledge is required. The ADBench [Han *et al.*, 2022] paper highlights the importance of labeled data in anomaly detection tasks, demonstrating that the performance of anomaly detection models is highly dependent on the availability of high-quality labeled instances. This finding underlines the need for effective strategies to obtain and utilize labeled data efficiently.

Active learning is an approach that addresses this challenge by intelligently selecting the most informative instances for labeling, thereby reducing the labeling cost while maintaining high detection performance. Despite its potential benefits, the application of active learning to anomaly detection tasks has not been extensively explored, and there is a lack of comprehensive evaluations and pipelines for active anomaly detection. In this research project, we aim to bridge this gap by conducting a systematic investigation of various active learning methods applied to anomaly detection tasks. Our study evaluates the performance, strengths, and weaknesses of these methods under different conditions, with a focus on understanding the impact of outliers on the active learning strategies, providing insights into the potential of active learning research in enhancing anomaly detection accuracy and efficiency.

We summarize our main contributions as follows:

- The development of an active anomaly detection pipeline, facilitating further research and applications of active learning in the anomaly detection domain.
- A comprehensive evaluation and comparison of different active learning methods applied to anomaly detection tasks on diverse datasets.
- An investigation of the impact of outliers on the performance of active learning strategies and the identification of effective outlier handling techniques.

2 Related Work

In this section, we discuss the related work in the field of tabular anomaly detection, active learning, and their combination for active anomaly detection.

Tabular Anomaly Detection Existing TAD algorithms can be divided into three groups by the availability of ground truth labels: supervised, semi-supervised, and unsupervised TAD. Shallow methods like IForest [Liu *et al.*, 2008] and LOF [Breunig *et al.*, 2000] have good performances on AD problems. LOF employs a density-based approach and is a measurement of how isolated an object is from the neighborhood. When it comes to deep methods, DevNet [Pang *et al.*, 2019] leverages autoencoders for feature learning and a separate deep neural network for classification. XGBOD [Zhao and Hryniewicki, 2018], on the other hand, is an ensemble method combining the strengths of XGBoost and k-Nearest Neighbors-based

Outlier Detection, offering an effective and interpretable solution for anomaly detection with a focus on handling imbalanced data and diverse feature spaces.

Active Learning Active learning is a subfield of machine learning that focuses on training models with limited labeled data by iteratively selecting the most informative samples for labeling. A comprehensive survey of active learning techniques can be found in [Settles, 2009]. Various querying strategies have been proposed, including uncertainty-based strategies, such as margin sampling [Scheffer *et al.*, 2001] and BALD sampling[Gal and Ghahramani, 2016], diversity-based strategies, such as BADGE sampling[Ash *et al.*, 2019], and hybrid strategies, such as Learning Loss for Active Learning [Yoo and Kweon, 2019]. These strategies aim to identify instances that, once labeled, will provide the most significant improvement in the model's performance.

Active Anomaly Detection The combination of active learning and anomaly detection has gained increasing attention, as it can reduce the annotation effort required to achieve satisfactory performance in identifying anomalies. [Liu *et al.*, 2014] proposed an active learning framework for one-class SVMs to perform anomaly detection. Zhou and Paffenroth [Zhou and Paffenroth, 2017] introduced an active learning method for anomaly detection based on matrix factorization. In more recent work, [Zha *et al.*, 2020] proposed a framework for active anomaly detection using deep learning models, which inspired our research.

3 ActiveAD Pipeline

In this section, we elaborate on the proposed active anomaly detection (ActiveAD) pipeline. An overview of ActiveAD is illustrated in Figure 1.

3.1 Data Handler

A Data handler is first designed to handle and manage labeled and unlabeled data for active anomaly detection task. The class has various methods to initialize, access, and manipulate labeled and unlabeled datasets. The primary functions of the class are:

- Store training and test datasets, along with their respective labels, and keep track of the labeled and unlabeled data in the training set.
- Initialize a given number or percentage of labeled data from the training dataset.
- Retrieve labeled, unlabeled, or partial datasets from the stored data.
- Calculate the test accuracy based on provided predictions and the metrics for anomaly detection, specifically the Area Under the Receiver Operating Characteristic curve (AUC-ROC) and the Area Under the Precision-Recall curve (AUC-PR).

3.2 Base Model Training

After wrapping the original data, the pipeline trains an initial anomaly detection base model using a small set of labeled instances. Any tabular anomaly detection model including supervised, semi-supervised, and unsupervised model can be

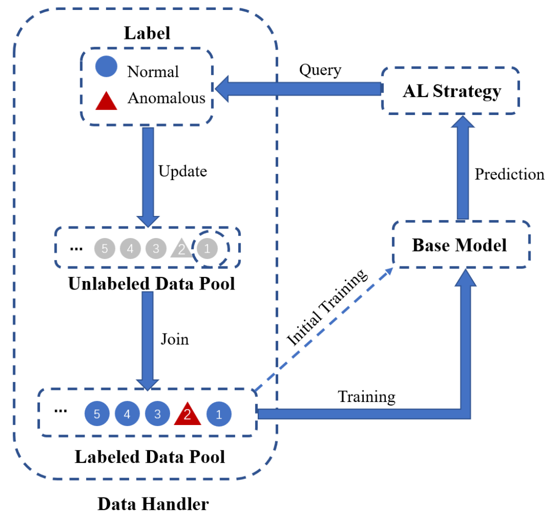


Figure 1: An overview of the proposed active anomaly detection pipeline.

chosen, but the choice should depend on the nature of the data and the specific requirements of the task. Ensure that the initial training set includes both normal and anomalous instances.

3.3 Active Learning Loop

The main part of the pipeline involves setting up the active learning loop. This typically consists of three stages: prediction, query, and update.

1. Prediction: Use the current model to predict the labels of the unlabeled instances.
2. Query: Select a subset of the unlabeled instances for labeling based on an active learning strategy. This could involve selecting the instances about which the model is most uncertain or those that are most informative or diverse.
3. Update: Add the labeled instances to the training set and update the model.

Repeat this loop until a stopping criterion is met, such as reaching a maximum number of iterations, using up the designated labels, or achieving a desired level of performance.

3.4 Active Learning Query Strategies

Based on different designs of query rules, we can classify current active learning strategies into three parts: Uncertainty-Based Strategies, Diversity-Based Strategies, and Hybrid Strategies. Most representative strategies will be introduced in detail in this section.

Margin Sampling

Margin Sampling [Scheffer *et al.*, 2001] is a naive active learning approach based on model uncertainty. It calculates a margin between the most and second most likely labels for one sample under the current model. The uncertainty is thus measured by the margin, i.e., lower margin means greater uncertainty. Specifically, the margin is calculated as:

$$\phi_M(x) = P_\theta(y_1^*|x) - P_\theta(y_2^*|x)$$

The sample points with smallest margins will be queried iteratively in the active learning process.

Bayesian Active Learning by Disagreement

Bayesian Active Learning by Disagreement [Gal and Ghahramani, 2016] is a more advanced uncertainty-based strategy. It aims to maximize the information gain (or Mutual Information), which is measured in terms of reduction in entropy of the posterior distribution of model parameters.

$$I(y;\boldsymbol{\omega}|x, D_{train}) = H(y|x, D_{train}) -$$

$$E_{\mathcal{P}(\boldsymbol{\omega}|D_{train})}[H(y|x,\boldsymbol{\omega}, D_{train})]$$

The larger the information gain, the greater the uncertainty.

BADGE Sampling

BADGE Sampling [Ash *et al.*, 2019] is a diversity-based AL strategy that focuses on the diversity of queried data. The main idea is to design an approach that creates a diverse batch of examples, about which the model is uncertain. The algorithm will first draw a random set of samples for initial training, then it will compute a gradient embedding for the samples with the initially trained model. According to the gradient embedding, it will use a clustering algorithm to select samples and query for their data. Detailed algorithm is shown in Appendix A.1

Learning Loss for Active Learning

Hybrid strategies like learning loss [Yoo and Kweon, 2019] are proposed in order to combine the advantages of both kinds of strategies. The idea of learning loss is to predict the loss of the target model, and query samples based on the predicted loss. It is assumed that samples with larger loss have greater uncertainty, meanwhile deviate to some extent with data in the current pool.

During training, the target prediction and the target annotation are used to compute a target loss to learn the target model as shown in Figure 2. Then, the target loss is regarded as a ground-truth loss for the loss prediction module, and used to compute the loss-prediction loss.

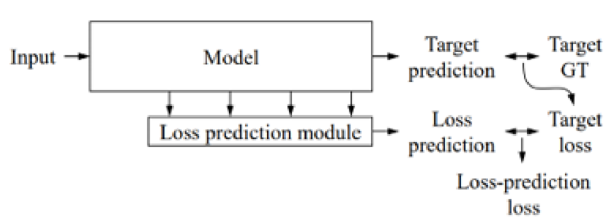


Figure 2: Loss Prediction Training

4 Experiments

4.1 Datasets

Proposed by Han et al., ADBench [Han *et al.*, 2022] is an open-source anomaly detection benchmark with 30 algorithms on over 50 datasets and extensive experiments. Various fields are covered in this benchmark suit, such as healthcare, audio and language processing, astronautics, image, and finance. To give a comprehensive and fair evaluation, we conduct experiments on 10 most unstable datasets [Audibert *et al.*, 2020] from ADBench. An unstable task means that the performance variance of different methods is large. For more details, please refer to the original paper.

4.2 TAD Base Model

DevNet [Pang *et al.*, 2019] and XGBOD [Zhao and Hryniewicki, 2018] are two popular and state-of-the-art machine learning models used for tabular anomaly detection. Both DevNet and XGBOD have been shown to be effective in detecting anomalies in a variety of settings. They can serve as strong backbone models when incorporating with different active learning algorithms.

4.3 Setup Details

In our experiments, we fix the initial label ratio as 10 percent and conduct experiments on 5 different budget ratios: 5 percent, 10 percent, 25 percent, 50 percent, and 75 percent of the total data size. The training batch size is set as one tenth of the budget size if the budget is larger than 320 or 32 otherwise.

5 Results and Analyses

We show the overall performance in Table 1. Due to the limited space, we only provide results under one setting in the main content. You can refer to more results in A.2.

Performance Comparisons

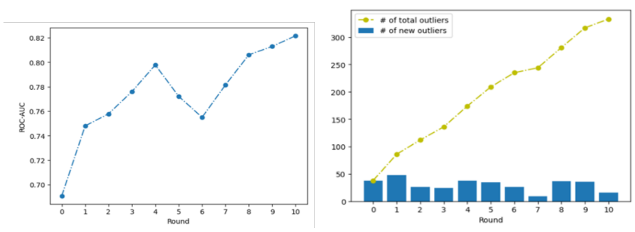
Given a fixed setting according to Table 1, we find that the active learning method can significantly improve the model performance compared to random sampling baseline. This demonstrate the effectiveness of active learning strategies under this ratio. However, when the budget ratio increase to 75 percent by Table 3, many strategies cannot beat random sampling. We can see that only the hybrid strategy that takes uncertainty and diversity both into account has better performance, showing the need for more complex strategy design in anomaly detection tasks.

Efficiency of Label Acquisition

The second observation can be drawn from Table 4 and Figure 3. When the process is divided into 10 rounds, we expect a a trend that the performance will keep increasing given more labels, and if more anomalies are found the performance would be better. However, we discover many unusual occurrences that the performance decreases as the model discovers more anomalies. This reveals several insights, such as the possibility that we may not always need to query the datapoints with the greatest anomaly score in each round. Instead, we may think about capturing the long-term performance of labelling

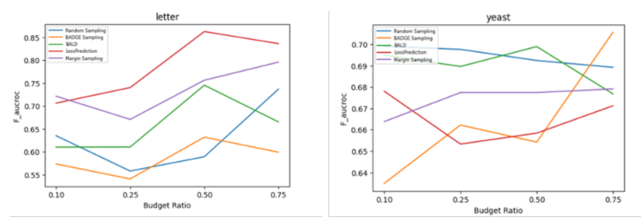
strategies	fault	internetads	ALOI	letter	magic	mammo	satellite	wave	yeast
RandomSampling	0.6919	0.7644	0.5112	0.5890	0.8581	0.8967	0.8471	0.8522	0.6924
AdversarialBIM	0.6467	0.8108	0.5681	0.5406	0.8029	0.9126	0.8255	0.5129	0.6906
AdversarialDF	0.6645	0.7708	**0.5851**	0.6927	0.8402	0.9126	0.8513	0.9316	0.6825
BALDDropout	0.7294	0.7942	0.5851	0.7453	0.8569	0.9276	0.8538	0.8721	**0.6989**
BadgeSampling	0.7149	0.8750	0.5432	0.6318	0.8629	0.9269	0.9180	0.8961	0.6542
EntropySampling	0.6692	0.7609	0.5831	0.7561	0.8410	0.9195	0.8390	0.9126	0.6774
EntropyDropout	0.6692	0.7609	0.5831	0.7561	0.8410	0.9195	0.8390	0.9126	0.6774
KCenterGreedy	0.7434	0.9232	0.5234	0.7332	0.8584	0.9277	0.8497	0.8860	0.6624
KCenterPCA	0.7542	0.9214	0.5376	0.7157	0.8614	0.8857	0.8526	0.5254	0.6586
KMeansSampling	0.6678	0.8472	0.5487	0.6390	0.8422	0.9290	0.8568	0.9170	0.6823
MarginSampling	0.6692	0.7609	0.5831	0.7561	0.8410	0.9195	0.8390	0.9126	0.6774
MarginDropout	0.6692	0.7609	0.5831	0.7561	0.8410	0.9195	0.8390	0.9126	0.6774
LossPrediction	**0.8215**	**0.9614**	0.5582	0.8624	**0.9108**	**0.9518**	**0.9769**	**0.9746**	0.6584
MeanSTD	0.7053	0.7912	0.5565	0.7607	0.8770	0.9275	0.8559	0.8883	0.6911
VarRatio	0.6692	0.7609	0.5831	0.8957	0.8410	0.9195	0.8390	0.9126	0.6774
WAAL	0.7850	0.9643	0.5728	**0.9804**	0.9040	0.9212	0.9494	0.9661	0.6610

Table 1: AUC-ROC of different query strategies, and baseline (random sampling) on all datasets. Base Model: DevNet. Budget Ratio: 0.5



(a) AUC-ROC performance (b) Number of outliers found

Figure 3: An intuitive illustration of single active learning process on fault dataset, the budget ratio is 0.5, the AL strategy is Loss Prediction (Base model: XGBOD).



(a) AL on letter (b) AL on yeast

Figure 4: Comparison between baseline strategy and AL strategies on one dataset (Fault), where budget ratio ranges from 0.05 to 0.75. (Base Model: DevNet).

in the design of active learning strategies, for instance, labelling anomalies first and then, after some iteration, labelling more informative datapoints like the border points.

Diminishing Returns

We notice that when the budget ratio increases from 5 percent to 75 percent, there are a increasing trend on the performance. However, we also notice that there are effect of diminishing returns for many datasets as shown in Figure 4. The reason may be as more data points are labeled, the model might have already learned most of the useful information from the data. Adding more labeled data after a certain point may not contribute much additional information, and the performance improvement becomes marginal or even negative. We further investigate into these marginal datasets and found that the number of features of these datasets are often large, this may inspire future practitioners to specifically design query strategies that can adapt to these high-dimensional data.

6 Significance of the Study

The significance of this study lies in its comprehensive investigation of various active learning methods applied to anomaly detection tasks, offering valuable insights into their performance, strengths, and weaknesses under varying conditions. Another essential contribution of this study is the focus on understanding the impact of outliers on the performance of active learning strategies. This is crucial for anomaly detection tasks, as outliers could significantly influence the detection models. The results can guide the development of robust active learning strategies and anomaly detection models that are capable of efficiently handling outliers.

7 Conclusion

The goal of this project was to understand the performance, strengths, and weaknesses of each AL method under different TAD conditions, with an emphasis on understanding the impact of outliers on these strategies. The proposed ActiveAD pipeline and extensive experiment results showed that AL methods have the potential to improve anomaly detection performance while reducing labeling cost. However, the performance of these methods varied depending on the specific characteristics of the datasets and the presence of outliers. Future research can build upon these findings, exploring alternative anomaly detection models, developing scalable active learning methods, and integrating additional sources of information.

References

[Ahmed *et al.*, 2016] Mohiuddin Ahmed, Abdun Naser Mahmood, and Md Rafiqul Islam. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, 55:278–288, 2016.

[Ash *et al.*, 2019] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*, 2019.

[Audibert *et al.*, 2020] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. Usad: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3395–3404, 2020.

[Breunig *et al.*, 2000] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[Djenouri *et al.*, 2018] Youcef Djenouri, Arthur Zimek, and Marco Chiarandini. Outlier detection in urban traffic flow distributions. In *2018 IEEE international conference on data mining (ICDM)*, pages 935–940. IEEE, 2018.

[Gal and Ghahramani, 2016] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

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

[Himeur *et al.*, 2021] Yassine Himeur, Khalida Ghanem, Abdullah Alsalemi, Faycal Bensaali, and Abbes Amira. Artificial intelligence based anomaly detection of energy consumption in buildings: A review, current trends and new perspectives. *Applied Energy*, 287:116601, 2021.

[Liu *et al.*, 2008] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[Liu *et al.*, 2014] Wei Liu, Gang Hua, and John R. Smith. Unsupervised one-class learning for automatic outlier removal. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3826–3833, 2014.

[Pang *et al.*, 2019] Guansong Pang, Chunhua Shen, and Anton van den Hengel. Deep anomaly detection with deviation networks, 2019.

[Riazi *et al.*, 2019] Mohammad Riazi, Osmar Zaiane, Tomoharu Takeuchi, Anthony Maltais, Johannes Günther, and Micheal Lipsett. Detecting the onset of machine failure using anomaly detection methods. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 3–12. Springer, 2019.

[Scheffer *et al.*, 2001] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *Advances in Intelligent Data Analysis: 4th International Conference, IDA 2001 Cascais, Portugal, September 13–15, 2001 Proceedings 4*, pages 309–318. Springer, 2001.

[Settles, 2009] Burr Settles. Active learning literature survey. 2009.

[Weller-Fahy *et al.*, 2014] David J Weller-Fahy, Brett J Borghetti, and Angela A Sodemann. A survey of distance and similarity measures used within network intrusion anomaly detection. *IEEE Communications Surveys & Tutorials*, 17(1):70–91, 2014.

[Yoo and Kweon, 2019] Donggeun Yoo and In So Kweon. Learning loss for active learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 93–102, 2019.

[Zha *et al.*, 2020] Daochen Zha, Kwei-Herng Lai, Mingyang Wan, and Xia Hu. Meta-aad: Active anomaly detection with deep reinforcement learning, 2020.

[Zhao and Hryniewicki, 2018] Yue Zhao and Maciej K. Hryniewicki. XGBOD: Improving supervised outlier detection with unsupervised representation learning. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2018.

[Zhou and Paffenroth, 2017] Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 665–674, 2017.

A Appendix

A.1 BADGE Algorithm

The main idea of BADGE Sampling [Ash *et al.*, 2019] is to design an approach that creates a diverse batch of examples, about which the model is uncertain.

Algorithm: Batch Active Learning by Diverse Gradient Embeddings

---

Input: Neural network $f(x;\theta)$, unlabeled pool of examples U, initial number of examples M, number of iterations T, number of examples in a batch B.

1: Labeled dataset $S \leftarrow M$ examples drawn uniformly at random from U together with queried labels.
2: Train an initial model θ_1 on S by minimizing loss.
3: **for** $t = 1, 2, \ldots, T$: **do**
4: For all examples x in $U \backslash S$:
  1. Compute its hypothetical label $\hat{y}(x)$
  2. Compute gradient embedding $g_x = \frac{\partial}{\partial \theta_{out}} \ell_{CE}(f(x;\theta), \hat{y}(x))$, where θ_{out} refers to parameters of the final (output) layer
5: Compute S_t, a random subset of $U \backslash S$ using the k-MEANS++ seeding algorithm on $\{g_x : x \in U \backslash S\}$ and query for their labels.
6. $S \leftarrow S \cup S_t$
7. Train a model θ_{t+1} on S by minimizing loss.
8. **end for**
9. **return** Final model θ_{T+1}.

A.2 Experimental Results

Experimental results for different budget ratios are shown in the Table 2 and Table 3. Table 4 show the single active learning process which incorporates the discussion in Section 5

strategies	fault	internetads	ALOI	letter	magic	mammo	satellite	wave	yeast
RandomSampling	0.6178	0.6301	0.5395	0.5578	0.8408	0.9056	0.8351	0.8522	**0.6976**
AdversarialBIM	0.6490	0.6647	0.5428	0.5734	0.7786	0.9125	0.8089	0.5000	0.6967
AdversarialDF	0.6408	0.6911	**0.5842**	0.5711	0.7961	0.9127	0.8449	0.9258	0.6649
BALDDropout	0.6838	0.6608	0.5674	0.6103	0.8719	0.9117	0.8515	0.8883	0.6896
BadgeSampling	0.6395	0.6750	0.5539	0.5406	0.8747	0.9224	0.8504	0.7926	0.6622
EntropySampling	0.6374	0.6402	0.5826	0.6706	0.7911	0.9169	0.8447	0.9095	0.6774
EntropyDropout	0.6374	0.6402	0.5826	0.6706	0.7911	0.9169	0.8447	0.9095	0.6774
KCenterGreedy	0.6854	0.8577	0.5006	0.6041	0.8215	**0.9297**	0.8477	0.8660	0.6847
KCenterPCA	0.6334	0.8200	0.5320	0.7304	0.8271	0.8706	0.8438	0.5419	0.6580
KMeansSampling	0.6033	0.7108	0.5585	0.5584	0.8267	0.9205	0.8503	0.5331	0.6512
MarginSampling	0.6374	0.6402	0.5826	0.6706	0.7911	0.9169	0.8447	0.9095	0.6774
MarginDropout	0.6374	0.6402	0.5826	0.6706	0.7911	0.9169	0.8447	0.9095	0.6774
LossPrediction	**0.7631**	**0.9560**	0.5532	**0.7401**	**0.9090**	0.9233	**0.9598**	**0.9755**	0.6533
MeanSTD	0.6782	0.6618	0.5469	0.6424	0.8787	0.9182	0.8567	0.8891	0.6546
VarRatio	0.6374	0.6402	0.5826	0.6706	0.7911	0.9169	0.8447	0.9095	0.6774
WAAL	0.7340	0.9529	0.5741	0.6616	0.9036	0.9050	0.8928	0.9661	0.6705

Table 2: AUC-ROC of different query strategies, and baseline (random sampling) on all datasets. Base Model: DevNet. Budget Ratio: 0.25

strategies	fault	internetads	ALOI	letter	magic	mammo	satellite	wave	yeast
RandomSampling	0.7520	0.8635	0.5645	0.7367	0.8642	0.9172	0.8552	0.8985	0.6892
AdversarialBIM	0.6578	0.8419	0.5622	0.5657	0.8412	0.9133	0.8488	0.5536	0.6807
AdversarialDF	0.6615	0.8520	0.5799	0.6294	0.8554	0.9276	0.8531	0.9182	0.6876
BALDDropout	0.7056	0.8269	0.5578	0.6651	0.8582	0.9277	0.8553	0.9029	0.6768
BadgeSampling	0.7388	0.8532	0.5393	0.5990	0.8668	0.9312	0.9330	0.9166	0.7056
EntropySampling	0.6769	0.8664	0.5869	0.7957	0.8451	0.9275	0.8498	0.9203	0.6791
EntropyDropout	0.6769	0.8664	0.5869	0.7957	0.8451	0.9275	0.8498	0.9203	0.6791
KCenterGreedy	0.7731	0.9406	0.5349	0.6979	0.8749	0.9278	0.8542	0.9217	0.6937
KCenterPCA	0.7718	0.9413	0.5484	0.7475	0.8731	0.9183	0.8519	0.7248	**0.7039**
KMeansSampling	0.7252	0.9321	0.5561	0.7097	0.8636	0.9292	0.8571	0.9415	0.6817
MarginSampling	0.6769	0.8664	0.5869	0.7957	0.8451	0.9275	0.8509	0.9203	0.6791
MarginDropout	0.6769	0.8664	0.5869	0.7957	0.8451	0.9275	0.8509	0.9203	0.6791
LossPrediction	**0.8315**	**0.9677**	0.5663	0.8361	**0.9164**	**0.9582**	**0.9795**	**0.9753**	0.6712
MeanSTD	0.6970	0.8742	0.5663	**0.9277**	0.8719	0.9275	0.8568	0.9753	0.6825
VarRatio	0.6769	0.8664	**0.5869**	0.7957	0.9275	0.9195	0.8509	0.9203	0.6791
WAAL	0.8012	0.9645	0.5364	0.7883	**0.9306**	0.9212	0.9505	0.9398	0.6664

Table 3: AUC-ROC of different query strategies, and baseline (random sampling) on all datasets. Base Model: DevNet. Budget Ratio: 0.75

	All X	All Outliers	#Batch	New Outliers	ROC-AUC
Round_0	136	38	68	38	0.6907
Round_1	204	86	68	48	0.7479
Round_2	272	112	68	26	0.7578
Round_3	340	136	68	24	0.7759
Round_4	408	174	68	38	0.7977
Round_5	476	209	68	35	0.7721
Round_6	544	235	68	26	0.7549
Round_7	612	244	68	9	0.7814
Round_8	680	281	68	37	0.8060
Round_9	748	317	68	36	0.8128
Round_10	816	333	68	16	0.8215

Table 4: A single active learning process on fault dataset, Base Model: XGBOD. Budget Ratio: 0.5. AL Strategy: Loss Prediction.