Pareto Optimization for Active Learning under Out-of-Distribution Data Scenarios

Xueying Zhan[*] *xyzhan2-c@my.cityu.edu.hk*
City University of Hong Kong

Zeyu Dai[†] *ze-yu.dai@connect.polyu.hk*
The Hong Kong Polytechnic University

Qingzhong Wang *qingzwang@outlook.com*
Baidu Research

Qing Li *csqli@comp.polyu.edu.hk*
The Hong Kong Polytechnic University

Haoyi Xiong *haoyi.xiong.fr@ieee.org*
Baidu Research

Dejing Dou *dejingdou@gmail.com*
BCG X

Antoni B. Chan *abchan@cityu.edu.hk*
City University of Hong Kong

Reviewed on OpenReview: *https://openreview.net/forum?id=dXnccpSSYF*

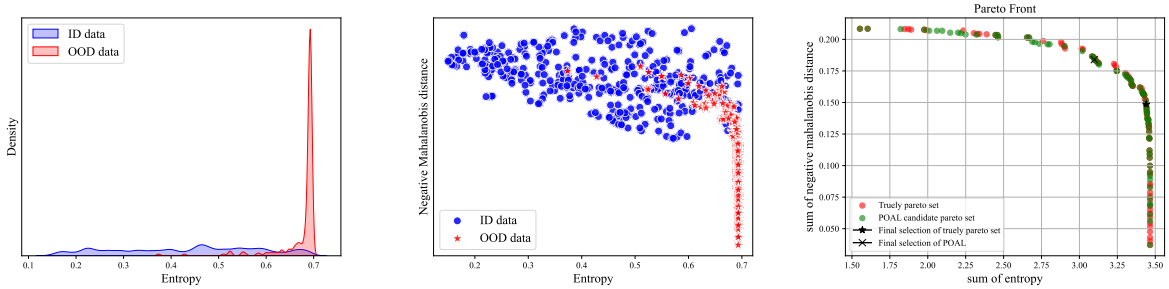
Abstract

Pool-based Active Learning (AL) has proven successful in minimizing labeling costs by sequentially selecting the most informative unlabeled data from large pool and querying their labels from an oracle or annotators. However, existing AL sampling schemes may not perform well in out-of-distribution (OOD) data scenarios, where the unlabeled data pool contains samples that do not belong to the pre-defined categories of the target task. Achieving strong AL performance under OOD data scenarios presents a challenge due to the inherent conflict between AL sampling strategies and OOD data detection. For instance, both more informative in-distribution (ID) data and OOD data in an unlabeled data pool would be assigned high informativeness scores (e.g., high entropy) during AL processes. To address this dilemma, we propose a Monte-Carlo Pareto Optimization for Active Learning (POAL) sampling scheme, which selects optimal subsets of unlabeled samples with *fixed batch size* from the unlabeled data pool. We formulate the AL sampling task as a multi-objective optimization problem and employ Pareto optimization based on two conflicting objectives: (1) the conventional AL sampling scheme (e.g., maximum entropy) and (2) the confidence of excluding OOD data samples. Experimental results demonstrate the effectiveness of our POAL approach on classical Machine Learning (ML) and Deep Learning (DL) tasks.

1 Introduction

In practical applications, obtaining large amounts of unlabeled data is relatively easy, but the process of labeling them can be expensive and time-consuming (Shen et al., 2004). AL aims to solve this problem – it

[*]Work is completed while Xueying Zhan is at Baidu Research, Big Data Lab.
[†]The first two authors Xueying Zhan and Zeyu Dai are equally contributed.



(a) Entropy score of ID and OOD data (b) Scatter plot of ID and OOD data (c) POAL vs. True PO

Figure 1: (a) Entropy distribution for ID and OOD data during AL processes on *EX8* dataset. (b) Scatter plot of the AL score (entropy) and ID confidence score (negative Mahalanobis distance) of unlabeled data. A larger ID score indicates that data is more likely to be ID data. (c) An example of the difference in the selection between our POAL and True PO (POAL uses Monte-Carlo for approximation, while True PO makes final selection after transverse the whole search space) on *EX8* dataset, where each element of Pareto set is a selected subset with fixed batch size.

achieves higher model performance with less training data by sequentially selecting the most informative instances and then querying their labels from oracles/annotators (Zhan et al., 2021b). However, current AL methods have primarily been evaluated on well-studied, simple, and clean datasets (Kothawade et al., 2021) like *MNIST* (Deng, 2012), *CIFAR10* (Krizhevsky et al., 2009), etc. However, in real-life scenarios, when collecting unlabeled data, it is common for task-unrelated data, referred to as out-of-domain (OOD) data, to be mixed in with the task-related data. For instance, when the task involves classifying images of digits, images of letters might also be present in the unlabeled data pool (Du et al., 2021). Most AL methods are not robust to OOD data scenarios. For example, Karamcheti et al. (2021) has demonstrated empirically that collective outliers hurt AL performances under Visual Question Answering (VQA) tasks. Furthermore, selecting and querying OOD samples that are irrelevant to the target task can lead to wasted labeling costs and reduce the effectiveness of the AL sampling process(Du et al., 2021). In this paper, we focus on the latter type of OOD data during AL, where the unlabeled data pool contains instances that do not belong to the classes of the target classification task.

There is a natural conflict between AL sampling process and OOD data detection. Most AL methods, especially uncertainty-based measures, prefer selecting data that are hardest to be classified by the current basic learner, e.g., high entropy of predicted class probabilities. However, in AL, if a basic learner (e.g., Neural Network with softmax output) performs poorly on ID data, it is more likely to provide non-informative predicted probabilities (i.e., close to uniform probabilities) on OOD data (Vaze et al., 2021). During AL processes, the basic learner is not well-trained due to the insufficient labeled data, and insufficient epochs in the case of deep AL. Therefore, the samples selected by AL may contain high informative ID and OOD samples. For instance, consider the Maximum Entropy (ENT) approach for AL, which is a classic uncertainty-based method (Lewis & Catlett, 1994; Shannon, 2001) that selects data samples whose predicted class probabilities have the largest entropy. Meanwhile, ENT is *also a typical OOD detection method* – high entropy of the predicted class distribution suggests that the input may be OOD (Ren et al., 2019). Fig. 1a shows an example on *EX8* data (Ng, 2008), which further illustrates this conflict, that is, a large percentage of data with high entropy scores are OOD, and thus ENT-based AL will likely to select OOD data for labeling. Thus, additional measures are needed to detect OOD samples so that they are not selected for AL. Fig. 1b is an example shows that the negative Mahalanobis distance (Lee et al., 2018) has a certain negative correlation with entropy and thus could be used as an ID confidence score.

Although the OOD problem has been demonstrated to affect AL in realistic scenarios (Karamcheti et al., 2021), there are only a few studies about it (Kothawade et al., 2021; Du et al., 2021). SIMILAR (Submodular Information Measures Based Active Learning) (Kothawade et al., 2021) adopted the submodular conditional mutual information (SCMI) function as the acquisition function. They jointly model the similarity between unlabeled and labeled ID data sets and their dissimilarity with labeled OOD data sets. The estimation

might not be accurate initially since labeled ID and OOD data sets are insufficient in the early stages of AL. Additionally, calculating SCMI on large-scale data is time/memory-consuming. CCAL (Contrastive Coding AL) (Du et al., 2021) needs to pre-train extra self-supervised models like SimCLR (Chen et al., 2020), and also introduces hyper-parameters to trade-off between semantic and distinctive scores, whose values affect the final performance (see Section 4.3 in (Du et al., 2021)). These two factors limit the range of its application.

In this paper, we advocate simultaneously considering the AL criterion and ID confidence when designing AL sampling strategies to address the above issues. Since the two objectives conflict, we define the AL sampling process under OOD data scenarios as a multi-objective optimization problem (Seferlis & Georgiadis, 2004). Unlike traditional methods for handling multiple-criteria-based AL, such as weighed-sum optimization (Zhan et al., 2022a) or two-stage optimization (Shen et al., 2004; Zhan et al., 2022a), we propose a novel and flexible batch-mode **P**areto **O**ptimization **A**ctive **L**earning (POAL) framework. The contributions and summarization of this paper are as follows:

1. We propose AL under OOD data scenarios within a multi-objective optimization framework.
2. Our framework is flexible and can accommodate different combinations of AL and OOD detection methods according to various target tasks. Our experiments use ENT as the AL objective and Mahalanobis distance as ID confidence scores.
3. Naively applying Pareto optimization to AL will result in a Pareto Front with a non-fixed size, which can introduce high computational costs. To enable efficient Pareto optimization, we propose a *Monte-Carlo (MC) Pareto optimization algorithm for fixed-size batch-mode AL*. MCPOAL could also provide near-optimal Pareto curves, as shown in Fig. 1c.
4. Our framework works well on classical ML and DL tasks. We propose pre-selecting and early-stopping techniques to reduce the computational cost of large-scale datasets.
5. Our framework has no trade-off hyper-parameter for balancing AL and OOD objectives. It is crucial since i) AL is data-insufficient, there might be no validation set for tuning parameters; ii) hyper-parameter tuning in AL can be label-expensive since every change of hyper-parameter causes AL to label new data, thus provoking substantial labeling inefficiency (Ash et al., 2020).

2 Related Work

We discuss the related work in the following categories separately. We briefly introduced pool-based AL, OOD and distribution shift data problems, how OOD data influence AL processes and multi-objection optimization.

2.1 Pool-based Active Learning

Pool-based AL has been well-studied in recent years (Settles, 2009; Zhan et al., 2021b; Ren et al., 2021; Zhan et al., 2022a) and widely adopted in various tasks (Duong et al., 2018; Yoo & Kweon, 2019; Dor et al., 2020; Haussmann et al., 2020). Most AL methods rely on fixed heuristic sampling strategies, which follow two main branches: uncertainty- and representative/diversity-based measures (Ren et al., 2021; Zhan et al., 2022a). Uncertainty-based approaches select data that maximally reduce the uncertainty of the target basic learner (Ash et al., 2020). Typical uncertainty-based measures that perform well on classical ML tasks like Query-by-Committee (QBC) (Seung et al., 1992), Bayesian Active Learning by Disagreement (BALD) (Houlsby et al., 2011) have also been generalized to DL tasks (Wang & Shang, 2014; Gal et al., 2017; Beluch et al., 2018; Zhan et al., 2022a). Representative/diversity-based methods like k-Means (Zhan et al., 2022a) and Core-Set approach (Sener & Savarese, 2018) select a batch of unlabeled data most representative of the set. Uncertainty/representative-based measures could be combined with weighted-sum or multi-stage optimization (Zhan et al., 2022a). Weighted-sum optimization combines multiple objectives through linear combination with trade-off weights. (Yin et al., 2017) is a typical method that adopts weighted-sum optimization, selecting the most uncertain and least redundant samples as well as the most diverse. Two-stage (or multi-stage) optimization first ranks data or selects a subset based on one objective, then makes final decisions based on the other objective (Shen et al., 2004; Zhao et al., 2019; Zhan et al., 2022a). (Ash et al., 2020) computes gradient embedding of unlabeled data in the first stage (uncertainty), then clusters by KMeans++ in the second stage (diversity).

2.2 Out-of-Distribution Data and Distribution Shift

OOD data detection. Detecting OOD data is vital in ensuring the reliability and safety of ML systems in real-life applications (Yang et al., 2021) since the OOD problem severely influences real-life decisions. For example, in medical diagnosis, the trained classifier could wrongly classify a healthy OOD sample as pathogenic (Ren et al., 2019; Cui et al., 2020). Existing methods compute ID/OOD confidence scores based on the predictions of (ensembles of) classifiers trained on ID data, e.g., the ID confidence can be the entropy of the predictive class distribution (Ren et al., 2019). Hendrycks & Gimpel (2017) observed that a well-trained neural network assigns higher softmax scores to ID data than OOD data, i.e., predictions for ID data have lower entropy. Follow-up work ODIN (Liang et al., 2017) amplifies the effectiveness of OOD detection with softmax score by considering temperature scaling and input pre-processing. Others (Lee et al., 2018) propose simple and effective methods for detecting both OOD and adversarial data samples based on Mahalanobis distance. Due to its superior performance compared with other OOD strategies (see Table 1 in (Ren et al., 2019)), our POAL framework uses Mahalanobis distance as the primary criterion for calculating the ID confidence score. Nonetheless, our framework is general, and any ID confidence score could be adopted.

How OOD influences AL methods? As discussed in Section 1, uncertainty-based AL measures like ENT are not robust to OOD data scenarios since OOD data are naturally difficult to be classified. Representative/diversity-based methods are also not robust to OOD data since outliers/OOD samples are far from ID data and thus more likely to be selected first to cover the whole unlabeled data pool. Combined AL strategies that integrate uncertainty and representative-based measures will also be susceptible to OOD data. Weighted-sum optimization will select data with high uncertainty and representativeness scores, which are likely to be OOD. The multi-stage optimization adopts uncertainty-based measures to select a subset and then uses representative-based measures for the final selection (or vice versa). For both variants, a large proportion of OOD data will be selected in the first stage. To address these issues, we proposed our POAL, which can be adapted to various AL sampling schemes with OOD data. We conducted experiments on multiple highly-cited AL methods to validate our viewpoints, as shown in Section 4, Fig. 4.

Distribution shift. AL with OOD data scenarios and AL with distribution shift scenarios share a common characteristic of a gap between the true underlying data distribution and the estimated distribution from labeled and unlabeled data (Zhan et al., 2022b). The main difference is in the specific nature of the distribution gap. Distribution shift (aka dataset shift), refers to the discrepancy between the data distributions of the training and testing sets (or true underlying data distribution). There are several branches of AL with distribution shift: 1) where both labeled and unlabeled data are from the same domain but the labeled set is not sampled independently and identically distributed (*i.i.d.*) from the true data distribution due to the data collection process in AL, often addressed using fixed heuristics for data sampling (Beygelzimer et al., 2009; Sawade et al., 2010; Ganti & Gray, 2012; Farquhar et al., 2021; Zhan et al., 2022b); and 2) where the distribution shift occurs between the unlabeled data pool and the target tasks, such as when the target task is to classify Domestic Shorthair/Longhair cats while the unlabeled data pool contains various types of cats. In the former case, some works model the discrepancy between the labeled data and the true underlying data distribution, leveraging techniques like importance sampling to train unbiased basic learners. Other approaches focus on modeling the difference between the labeled and unlabeled sets or the full data pool to inform AL sampling strategies. For instance, Shui et al. (2020) utilizes unlabeled data information by training a discriminator to distinguish between labeled and unlabeled data sets.

In the second case, some works model the discrepancy between source and target domain of labeled/unlabeled data pools to select more target-task-related data samples. Mahmood et al. (2021) minimizes the Wasserstein distance between the unlabeled set and the set to be labeled as AL sampling strategy. de Mathelin et al. (2021) discusses AL under domain shift for general loss functions and provides a generalization bound of the target risk based on localized discrepancy distances between sample points. Zhao et al. (2021) presents an AL framework for addressing label shift problems by combining importance weighting and sub-sampling techniques. Distribution shifts can be further applied to train better basic classifiers (Imberg et al., 2020; Farquhar et al., 2021; Zhan et al., 2022b). In these AL approaches under distribution shift, distribution shift is useful for training AL. However, in AL with OOD data scenarios, OOD samples are hard to utilize for

model training since OOD samples are not in the classes of interest of the target classification task. Thus, the current research aims to detect OOD samples and avoid sampling them in the first place.

2.3 Multi-objective Optimization

Many real-life applications require optimizing multiple objectives that conflict with each other. For example, in the sensor placement problem, the goal is to maximize sensor coverage while minimizing deployment costs (Watson et al., 2004). Since there is no single solution that simultaneously optimizes each objective, Pareto optimization can be used to find a set of "*Pareto optimal*" solutions with optimal trade-offs of the objectives (Miettinen, 2012). A decision maker can then select a final solution based on their requirements. Compared with traditional optimization methods for multiple objectives (e.g., weighted-sum (Marler & Arora, 2010)), Pareto optimization algorithms are designed via different meta-heuristics *without any trade-off parameters* (Zhou et al., 2011; Liu et al., 2021). In our work, besides optimization of the two conflicting objectives (AL score and ID confidence), we also need to perform batch-mode subset selection. Pareto Optimization Subset Selection (POSS) (Qian et al., 2015) solves the subset selection problem by optimizing two objectives simultaneously: maximizing the criterion function and minimizing the subset size. However, POSS does not support *more than one criterion and cannot work with fixed subset size*, thus not suitable for our task. Therefore, we propose Monte-Carlo POAL, which achieves: 1) optimization of multiple objectives; 2) no extra trade-off parameters for tuning; 3) subset selection with a fixed size.

3 Methodology

In this section we introduce the problem definition of AL under OOD data scenarios, and details of POAL.

3.1 Problem Definition and Overview

We consider a general pool-based AL process for a K-class classification task with feature space $\mathcal{X}$, label space $\mathcal{Y} \in \{1, ..., K\}$ under OOD data scenarios. We assume that the oracle can provide a fixed number of labels, denoted as budget B. When an OOD sample, which does not belong to one of the K classes, is queried, the oracle will return an "OOD" label[1] to represent data outside of the specified task. We aim to select the most informative B instances with fewer OOD samples to obtain the best classification performance. To reduce the computational cost, we consider batch-mode AL, where batches of samples with fixed size b are selected and queried. We denote the AL acquisition function as $\alpha(\mathbf{x}; \mathcal{A})$, where $\mathcal{A}$ refers to AL sampling strategy, and the basic learner/classifier as $f_\theta(\mathbf{x})$. We denote the current labeled set as $\mathcal{D}_l = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and the large unlabeled data pool as $\mathcal{D}_u = \{\mathbf{x}_i\}_{i=1}^M$. The labeled data are sampled *i.i.d.* over In-Distribution (ID) data $\mathcal{D}$, i.e., $\mathcal{D}_l \subset \mathcal{D}$, and $N \ll M$. Under OOD data scenarios, active learners may query OOD samples whose labels are not in $\mathcal{Y}$. To simplify the problem settings, we ignore the queried OOD data and only add ID samples to $\mathcal{D}_l$. Although OOD data could be used to improve model robustness,(Wei et al., 2021) demonstrated that this requires very large-scale OOD/noisy data ($>$ 160k samples). However, in the AL settings, the size of $\mathcal{D}_l$ is limited, especially for $\mathcal{D}_l^{\text{OOD}}$, and thus we do not consider using OOD data for training in this paper. Selecting high-informative ID samples while preventing OOD data selection benefits more the AL processes.

Motivated by the natural conflict between AL and OOD data detection, as illustrated in Fig. 1b, we propose POAL as outlined in Fig. 2. In each AL iteration, we firstly utilize the clean labeled set that only contains ID data for training a basic classifier $f_\theta(\mathbf{x})$ and constructing class-conditional Gaussian Mixture Models (GMMs) for detecting OOD samples in $\mathcal{D}_u$. Based on classifier $f_\theta(\mathbf{x})$, AL acquisition function $\alpha(\mathbf{x}; \mathcal{A})$ and the GMMs, we calculate the informativeness/uncertainty score $\mathcal{U}(\mathbf{x}_i)$ and ID confidence score $\mathcal{M}(\mathbf{x}_i)$ for each unlabeled sample $\mathbf{x}_i$ in $\mathcal{D}_u$. Let $\mathbf{s} \in \{0, 1\}^M$ be a binary vector whose element $s_i = 1$ indicates that the i-th unlabeled sample is selected, and $s_i = 0$ indicates unselected. In each AL stage, the multi-objective optimization goal is to find an optimal subset $\mathbf{s}^*$ with b selected samples, that simultaneously maximizes the informativeness/uncertainty and ID confidence score:

[1] In our implementation, the label of OOD data is -1.

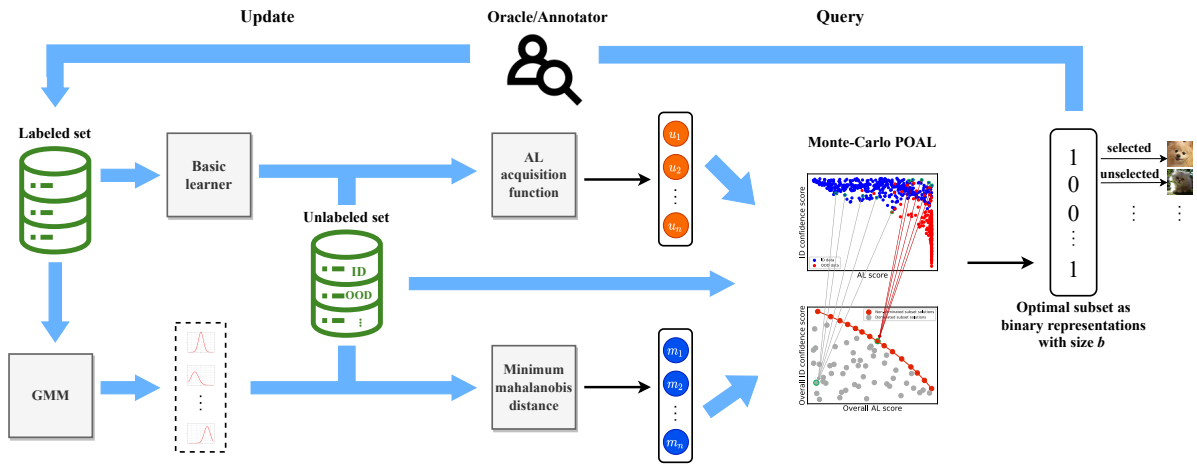


Figure 2: Overview of POAL. In each iteration, we train the basic learner and estimate GMMs (in DL tasks, we estimate multivariate Gaussian distributions with low-/high- feature levels) to calculate the AL and Mahalanobis distance-based ID confidence score for unlabeled data. We then construct POAL by randomly generating candidate solutions with fixed batch size and updating the Pareto set iteratively to get the optimal solution.

$$\mathbf{s}^* = \arg\max_{\mathbf{s}\in\{0,1\}^M} \; (\mathcal{U}(\mathbf{s}), \mathcal{M}(\mathbf{s})) \quad s.t. \quad \sum s_i = b, \tag{1}$$

where $\mathcal{U}(\mathbf{s}) = \sum_{i=1}^{M} s_i \mathcal{U}(\mathbf{x}_i)$ is shorthand for the aggregated AL score for the selected samples, and likewise for $\mathcal{M}(\mathbf{s})$. $\mathcal{U}(\cdot)$ and $\mathcal{M}(\cdot)$ will be introduced in Sections 3.2 and 3.3.

3.2 Active Learning Score

AL selects data by maximizing the acquisition function $a(\mathbf{x}; \mathcal{A})$, i.e., $\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{D}_u} a(\mathbf{x}; \mathcal{A})$ (Gal et al., 2017). In our work, we use the AL scores for the whole unlabeled pool $\mathcal{D}_u$ for the subsequent Pareto optimization. Thus, inspired by (Fredlund et al., 2010; Zhan et al., 2022b), we convert the acquisition function to a *querying density function* via $\mathcal{U}(\mathbf{x}_i) = \frac{\alpha(\mathbf{x}_i;\mathcal{A})}{\sum_{j=1}^{M} \alpha(\mathbf{x}_j;\mathcal{A})}$. In our experiments, we use ENT as our basic AL sampling strategy, and the acquisition function is $\alpha_{\text{ENT}}(\mathbf{x}) = -\sum_{k=1}^{K} p_\theta(y=k|\mathbf{x}) \log p_\theta(y=k|\mathbf{x}))$, where $p_\theta(y|\mathbf{x})$ is the posterior class probability using classifier $f_\theta(\mathbf{x})$. Our framework is flexible because it can incorporate various AL sampling strategies if their acquisition function can convert to a querying density function. In general, AL methods that explicitly provide per-sample scores (e.g., class prediction uncertainty) or inherently provide pair-wise rankings among the unlabeled pool (e.g., k-Means) can convert to a querying density. Thus, many uncertainty-based measures like QBC (Seung et al., 1992), BALD (Houlsby et al., 2011; Gal et al., 2017), and Loss Prediction Loss (LPL) (Yoo & Kweon, 2019) are applicable since they explicitly provide uncertainty information per sample. Also, k-Means and Core-Set (Sener & Savarese, 2018) approaches are applicable since they provide pair-wise similarity information for ranking. On the other hand, AL approaches that only provide overall scores for the candidate subsets, such as Determinantal Point Processes (Bıyık et al., 2019; Zhan et al., 2021a), are unable to be adopted in our framework.

3.3 In-Distribution Confidence Score via Mahalanobis Distance

Intuitively, ID unlabeled data can be distinguished from OOD unlabeled data since the ID unlabeled data should be closer to the ID labeled data ($\mathcal{D}_l$) in the feature space. One possible solution is to calculate the minimum distance between an unlabeled sample and the labeled data of its predicted pseudo label provided by $f(\theta)$ (Du et al., 2021). If this minimum distance is large, the unlabeled sample will likely be OOD and vice versa. However, calculating pair-wise distances between all labeled and unlabeled data is computationally expensive. A more efficient method is to summarize the labeled ID data via a data distribution (probability density function) and then compute the distances of the unlabeled data to the ID distribution. Mukhoti et al.

(2021) adopts a GMM as generative classifier to estimate the feature-space density, which is further used to capture epistemic uncertainty. Using GMMs on ID data to calculate ID confidence score for OOD detection in AL tasks has been shown to be a feasible solution in (Mukhoti et al., 2021). Furthermore, motivated by (Lee et al., 2018), we adopt GMMs to represent the data distribution, and then our ID confidence score is based on Mahalanobis distance. AL for classical ML tasks focuses on training basic learner $f_\theta(\mathbf{x})$ with fixed feature representations, while AL for DL jointly optimizes the feature representation $\mathcal{X}$ and classifier $f_\theta(\mathbf{x})$ simultaneously (Ren et al., 2021). Considering that these differences can influence the data distributions and further influence the Mahalanobis distance calculations, we adopt different settings for classical ML and DL.

Classical ML tasks. We represent the data distribution estimated from $\mathcal{D}_l$ with GMMs. Specifically, since we have labels of $\mathcal{D}_l$, we represent each class k with a class-conditional GMM,

$$p_{\text{GMM}}(\mathbf{x}|y=k) = \sum\nolimits_{c=1}^{C_k} \pi_c^{(k)} \mathcal{N}(\mathbf{x}|\mu_c^{(k)}, \Sigma_c^{(k)}), \tag{2}$$

where $(\pi_c^{(k)}, \mu_c^{(k)}, \Sigma_c^{(k)})$ are the mixing coefficient, mean vector and covariance matrix of the c-th Gaussian component with the k-th class, and $\mathcal{N}(\mathbf{x}|\mu, \Sigma)$ is a multivariate Gaussian distribution with mean μ and covariance Σ. The class-conditional GMM for class k is estimated from the labeled data for class k, $\mathcal{D}_l^{(k)} = \{\mathbf{x}_i | y_i = k\}$ using a maximum likelihood estimation (Reynolds, 2009) and the EM algorithm (Dempster et al., 1977). We then define the ID confidence score $\mathcal{M}(\mathbf{x})$ as the Mahalanobis distance between the unlabeled sample and its closest Gaussian component in any class,

$$\mathcal{M}_{\text{ML}}(\mathbf{x}) = \max_k \max_c -||\mathbf{x} - \mu_c^{(k)}||^2_{\Sigma_c^{(k)}}. \tag{3}$$

DL tasks. For DL, we follow (Lee et al., 2018) to calculate $\mathcal{M}(\mathbf{x})$, since it makes good use of both low- and high-level features generated by the deep neural network (DNN) and also applies calibration techniques, e.g., input-preprocessing (Liang et al., 2018) and feature ensembles Lee et al. (2018) to improve the OOD detection capability. Specifically, denote $f_\ell(\mathbf{x})$ as the output of the ℓ-th layer of a DNN for input $\mathbf{x}$. A class-conditional Gaussian distribution with shared covariance is estimated for each layer and for each class k by calculating the empirical mean and *shared* covariance $(\hat{\mu}_\ell^{(k)}, \hat{\Sigma}_\ell)$:

$$\hat{\mu}_\ell^{(k)} = \tfrac{1}{|\mathcal{D}_l^{(k)}|} \sum\nolimits_{\mathbf{x} \in \mathcal{D}_l^{(k)}} f_\ell(\mathbf{x}), \quad \hat{\Sigma}_\ell = \tfrac{1}{|\mathcal{D}_l|} \sum\nolimits_k \sum\nolimits_{\mathbf{x} \in \mathcal{D}_l^{(k)}} (f_\ell(\mathbf{x}) - \hat{\mu}_\ell^{(k)})(f_\ell(\mathbf{x}) - \hat{\mu}_\ell^{(k)})^T. \tag{4}$$

Next we find the closest class for each layer ℓ via $\hat{k}_\ell = \arg\min_k ||f_\ell(\mathbf{x}) - \hat{\mu}_\ell^{(k)}||^2_{\Sigma_\ell}$. To make the ID and OOD data more separable, an input pre-processing step is applied by adding a small perturbation to $\mathbf{x}$, $\hat{\mathbf{x}} = \mathbf{x} + \varepsilon \text{sign}(\nabla_{\mathbf{x}} ||f_\ell(\mathbf{x}) - \hat{\mu}_\ell^{(\hat{k}_\ell)}||^2_{\hat{\Sigma}_\ell})$, where ε controls the perturbation magnitude[2]. Finally, the ID confidence score is the average Mahalanobis distance for $\hat{\mathbf{x}}$ over the L layers,

$$\mathcal{M}_{\text{DL}}(\mathbf{x}) = \frac{1}{L} \sum\nolimits_{\ell=1}^{L} \max_k -||f_\ell(\hat{\mathbf{x}}) - \hat{\mu}_\ell^{(k)}||^2_{\hat{\Sigma}_\ell}. \tag{5}$$

Both AL and ID confidence score calculations will suffer from problems of insufficient training data. Insufficient training data is a common issue in AL, which can lead to bias in the AL score calculations. This bias problem is most prevalent in the initial AL iterations, as the insufficiently trained basic learners may provide inaccurate predictions. As the amount of training data increases, the bias problem caused by insufficient data gradually diminishes (Zhan et al., 2022b). For OOD detection, the Mahalanobis distance calculation is influenced by the capability of the basic classifier used as a feature extractor for estimating the Mahalanobis distance (in DL tasks), and the size of the ID labeled set. More data results in more accurate ID data distribution estimation, so ID/OOD confidence score calculation may also suffer from insufficient training data in the initial stages.

3.4 Monte-Carlo Pareto Optimization Active Learning

Our framework for AL for OOD data contains two criteria, AL score and ID confidence score. As discussed in Section 2, weighted-sum and multi(two)-stage optimization are widely used in multiple-criteria AL problems.

[2]In our experiments we follow (Lee et al., 2018) and set $\varepsilon = 0.01$.

However, both have drawbacks when applied to AL for the OOD data scenario. In weighted-sum optimization, the objective functions are summed up with weight η: $\alpha_{\text{WeightedSum}}(\mathbf{x}) = \eta\mathcal{U}(\mathbf{x}) + (1-\eta)\mathcal{M}(\mathbf{x})$, e.g., as adopted by CCAL (Du et al., 2021). This introduces an extra trade-off parameter η for tuning, but the true proportion of OOD data in $\mathcal{D}_u$ is unknown; thus, we cannot tell which criterion is more important. Furthermore, due to the lack of data, there is not enough validation data for properly tuning η. For two-stage optimization (Haneveld & van der Vlerk, 1999; Ahmed et al., 2004; Bindewald et al., 2020; Salo et al., 2022), a subset of possible ID samples is first selected with threshold $\mathcal{M}(\mathbf{x}) < \delta$, and then the most informative b samples in the subset are selected by maximizing $\mathcal{U}(\mathbf{x})$. However, suppose δ has not been properly selected. In that case, we might i) select OOD samples in the first stage (when δ is too large), and these OOD samples will be more likely to be selected first due to their higher AL score in the second stage; ii) select ID samples that are close to existing samples (when δ is too small), and due to the natural conflict between the two criteria, the resulting subset will be non-informative, thus influencing the AL performance (Zhao et al., 2019).

Considering this contradiction between AL and OOD criteria, it is vital to develop a combined optimization strategy that does not require manual tuning of this trade-off. Therefore, we propose POAL for balancing $\mathcal{U}(\mathbf{x})$ and $\mathcal{M}(\mathbf{x})$ automatically without requiring trade-off parameters. Consider one iteration of the AL process, where we must select b samples from $\mathcal{D}_u$. The search space size is the number of combinations M choose b, $\mathcal{C}(M, b)$, and the search is an NP-hard problem in general. Inspired by (Qian et al., 2015), we use a binary vector representation of candidate subset: $\mathbf{s} \in \{0,1\}^M$, where $s_i = 1$ represents i-th sample is selected, and $s_i = 0$ otherwise. For the unlabeled set $\mathcal{D}_u$, we denote the vector of AL scores as $\mathbf{u} = [\mathcal{U}(\mathbf{x}_i)]_{i=1}^M$ and ID confidence scores as $\mathbf{m} = [\mathcal{M}(\mathbf{x}_i)]_{i=1}^M$. The two criteria scores for the subset $\mathbf{s}$ are then computed as $o_U(\mathbf{s}) = \mathbf{s}^T\mathbf{u}$ and $o_M(\mathbf{s}) = \mathbf{s}^T\mathbf{m}$.[3] The ranking relationships between two candidate subsets $\mathbf{s}$ and $\mathbf{s}'$ are as follows:

- $\mathbf{s}' \preceq \mathbf{s}$ denotes that $\mathbf{s}'$ is *dominated* by $\mathbf{s}$, such that both scores for $\mathbf{s}'$ are no better than those of $\mathbf{s}$, i.e., $o_U(\mathbf{s}') \leq o_U(\mathbf{s})$ and $o_M(\mathbf{s}') \leq o_M(\mathbf{s})$.
- $\mathbf{s}' \prec \mathbf{s}$ denotes that $\mathbf{s}'$ is *strictly dominated* by $\mathbf{s}$, such that $\mathbf{s}'$ has one strictly smaller score (e.g., $o_U(\mathbf{s}') < o_U(\mathbf{s})$) and one score that is not better (e.g., $o_M(\mathbf{s}') \leq o_M(\mathbf{s})$).
- $\mathbf{s}$ and $\mathbf{s}'$ are *incomparable* if both $\mathbf{s}$ is not *dominated* by $\mathbf{s}'$ and $\mathbf{s}'$ is not *dominated* by $\mathbf{s}$.

Our goal (see Eq. 1) is to find an optimal subset solution $\mathbf{s}$ that is not dominated by other subset solutions,with $\sum s'_i = b$. The large search space makes traversing all possible subset solutions impossible. Thus we propose Monte-Carlo POAL for fixed-size subset selection. Monte-Carlo POAL iteratively generates a candidate solution $\mathbf{s}$ at random, and checks it against the current Pareto set $\mathcal{P} = \{\mathbf{s}_1, \mathbf{s}_2, \cdots\}$. If there are no candidate solution in $\mathcal{P}$ that *strictly dominates* $\mathbf{s}$, then $\mathbf{s}$ is added to $\mathcal{P}$ and all candidate solutions in $\mathcal{P}$ that are *dominated* by $\mathbf{s}$ are removed. In this way, a Pareto set $\mathcal{P}$ is maintained to include the solution(s) that dominate all other candidate solutions so far, while the solution(s) in $\mathcal{P}$ are incomparable with each other.

3.4.1 Early-stopping

We adopt an early-stopping strategy to automatically terminate POAL when there is no significant change in $\mathcal{P}$ after many successive iterations, which indicates that *a randomly generated* $\mathbf{s}$ *has little probability of changing* $\mathcal{P}$ since most non-dominated solutions are included in $\mathcal{P}$ (Saxena et al., 2016). Firstly, we need to define the difference between two Pareto sets, $\mathcal{P}$ and $\mathcal{P}'$. We represent each candidate solution $\mathbf{s}$ as a 2-dimensional feature vector, $v(\mathbf{s}) = (o_U(\mathbf{s}), o_M(\mathbf{s}))$. The two Pareto sets can then be compared via their feature vector distributions, $\mathbf{V}(\mathcal{P}) = \{v(\mathbf{s})\}_{\mathbf{s}\in\mathcal{P}}$ and $\mathbf{V}(\mathcal{P}') = \{v(\mathbf{s}')\}_{\mathbf{s}'\in\mathcal{P}'}$. We employed the maximum mean discrepancy (MMD) (Gretton et al., 2006) to measure the difference:

$$\text{MMD}(\mathcal{P}, \mathcal{P}') = \left\| \tfrac{1}{|\mathcal{P}|} \sum\nolimits_{\mathbf{s}\in\mathcal{P}} v(\mathbf{s}) - \tfrac{1}{|\mathcal{P}'|} \sum\nolimits_{\mathbf{s}'\in\mathcal{P}'} v(\mathbf{s}') \right\|_{\mathcal{H}}. \tag{6}$$

MMD is based on embedding probabilities in reproducing kernel Hilbert space $\mathcal{H}$, and here we use the RBF kernel with various bandwidths (i.e., multiple kernels). At iteration t, we compute the mean and standard deviation (SD) of the MMD scores in a sliding window of the previous s_w iterations,

$$M_t = \frac{1}{s_w} \sum\nolimits_{i=t-s_w+1}^{t} \text{MMD}(\mathcal{P}_{i-1}, \mathcal{P}_i), \quad S_t = \frac{1}{s_w} \sum\nolimits_{i=t-s_w+1}^{t} (\text{MMD}(\mathcal{P}_{i-1}, \mathcal{P}_i) - M_t)^2. \tag{7}$$

[3]To keep the two criteria consistent, we normalize the ID confidence scores: $\mathbf{m} \leftarrow \max(\mathbf{m}) - \mathbf{m}$.

If there is no significant difference in the mean and SD after several iterations, e.g., M_t and S_t do not change within two decimal points, then we assume that $\mathcal{P}$ has converged, and we stop iterating.

3.4.2 Final selection

After obtaining the converged set $\mathcal{P}$, we need to pick one $\mathbf{s} \in \mathcal{P}$ as the final solution (i.e., final subset for querying). Selecting a final solution is an open question in multi-objective optimization. Typical strategies (Chaudhari et al., 2010) to obtain a final solution include: (i) reformulate the problem as a single objective problem using additional information as required by the decision-makers, such as the relative importance or weights of the objectives, goal levels for the objectives, values functions, etc. (ii) Decision makers interact with the optimization procedure typically by specifying preferences between pairs of presented solutions. (iii) Output a representative set of non-dominated solutions approximating the Pareto front, e.g., regarding the candidate solutions as data points, performing clustering, and outputting the centroid as a final solution.

In our work, we regard candidate solutions as data points and use the cluster centroid as the final solution:

$$\mathbf{s}^* = \arg\min_{\mathbf{s}\in\mathcal{P}} \sum\nolimits_{\mathbf{s}'\in\mathcal{P}} \|\mathbf{s}' - \mathbf{s}\|^T = \arg\max_{\mathbf{s}\in\mathcal{P}} \sum\nolimits_{\mathbf{s}'\in\mathcal{P}} \mathbf{s}^T\mathbf{s}', \tag{8}$$

where we use the property $\mathbf{s}^T\mathbf{s} = \mathbf{s}'^T\mathbf{s}' = b$. Eq. 8) is equivalent to selecting the solution with a maximum intersection with other non-dominated solutions, i.e., we find the solution whose selected samples are commonly used in all other solutions in $\mathcal{P}$.

Each solution in $\mathcal{P}$ represents a particular ID-OOD trade-off since an appropriate choice of weight η in the weighted-sum objective will lead to its selection. Thus, the maximum intersection operation selects the samples that are *common to all settings* of the ID-OOD trade-off. In this sense, POAL bypasses the tuning of ID-OOD trade-off hyper-parameters by selecting samples that work well for all trade-offs.

POAL with early stopping and final selection strategies is summarized in Algorithm 1.

Algorithm 1 Monte-Carlo POAL with early stopping under OOD data scenarios.

Require: Unlabeled data pool $\mathcal{D}_u$ with size M, criteria $\mathcal{U}(\mathbf{x})$ and $\mathcal{M}(\mathbf{x})$, batch size b, maximum repeat time T, population interval p_{inv}, sliding window size s_w and final decision function $\mathcal{F}$.
Ensure: A subset $\mathcal{D}_s$ of $\mathcal{D}_u$ where $|\mathcal{D}_s| = b$.

```
Let s = {0}^M, P = {s} and t = 1.
for t in 1, ..., T do
    Generate a random solution s with condition Σ s_i = b.
    if ∄z ∈ P such that s ≺ z then
        Q = {z ∈ P | z ⪯ s}.
        P = (P \ Q) ∪ {s}.
    end if
    {Terminate the loop early if the pareto set P converged.}
    if t ≠ 0 then
        Calculate MMD_t according to Equation 6.
        if MOD(t, p_inv) = 0 then
            Calculate M_t and S_t with interval p_inv according to Equation 7 respectively[4].
            M̂_t = ROUND(M_t, 2), Ŝ_t = ROUND(S_t, 2).
            if M̂_t = M̂_{t-1} = ... = M̂_{t-s_w} and Ŝ_t = Ŝ_{t-1} = ... = Ŝ_{t-s_w} then
                Break
            end if
        end if
    end if
end for
return arg max_{s∈P} F(s).
```

3.4.3 Pre-selection for large-scale unlabeled data pool

We next consider how to use POAL on large datasets. The search space of size $\mathcal{C}(M, b)$ is vast for a large unlabeled data pool. We propose an *optional* pre-selection technique to decrease the search size by pre-selecting a subset of candidates. We select an optimistic subset $\mathcal{D}_{\text{sub}}$ from $\mathcal{D}_u$, based on the dominant relationships between different samples from the original unlabeled data pool. Firstly, an initial Pareto front $\mathcal{P}_{\text{pre}}$, a set of non-dominated data points, is selected according to two objectives $\mathcal{U}(\mathbf{x})$ and $\mathcal{M}(\mathbf{x})$. Since the size of $\mathcal{P}_{\text{pre}}$ is not fixed and might not meet our requirement of the minimum pre-selected subset size s_m, we iteratively update $\mathcal{D}_{\text{sub}}$ by firstly adding the data points in $\mathcal{P}_{\text{pre}}$ to $\mathcal{D}_{\text{sub}}$, and then excluding $\mathcal{D}_{\text{sub}}$ from $\mathcal{D}_u$ for the next round of selection. The iteration terminates when $|\mathcal{D}_{\text{sub}}| \geq s_m$. The pre-selected subset size s_m is set according to personal requirements (i.e., the computing resources and time budget). In our experiment, we set $s_m = 6b$. The pseudo-code of POAL with the pre-selection strategy is summarized in Algorithm 2.

Algorithm 2 Pre-selecting technique on large-scale datasets.

Require: Unlabeled data pool $\mathcal{D}_u$, criteria $\mathcal{U}$ and $\mathcal{M}$, minimum size of subset s_m of pre-selection.
Ensure: A pre-selected subset $\mathcal{D}_{\text{sub}}$ of $\mathcal{D}_u$ with $|\mathcal{D}_{\text{sub}}| \geq s_m$.

```
Let D_sub = ∅. i = 0.
while |D_sub| < s_m do
    Let D_p = ∅.
    for i in 0, ..., |D_u| do
        if ∄x ∈ D_p such that D_u^i ≺ x then
            Q = {x ∈ D_p | x ⪯ D_u^i}.
            D_p = (D_p \ Q) ∪ {D_u^i}.
        end if
    end for
    D_u = D_u \ D_p.
    D_sub = D_sub ∪ D_p.
end while
return D_sub.
```

3.4.4 POAL vs. POSS

Our Monte-Carlo POAL is inspired by Subset Selection by Pareto Optimization (POSS) (Qian et al., 2015). POSS selects a subset $S \subseteq V$ such that f is optimized with the constraint $|S| \leq k$, where $V = \{X_1, ..., X_n\}$ is a set of variables, f is a criterion function, and k is a positive integer. POSS is NP-hard in general (Davis et al., 1997). POSS has two conflicting objectives: maximizing the criterion function and minimizing the subset size. POSS first initializes Pareto set $\mathcal{P}$ with a random solution and then selects a solution from $\mathcal{P}$ to generate a new solution by flipping each bit with probability $1/n$. The new solution would add to $\mathcal{P}$ if it is not strictly dominated by any solution in $\mathcal{P}$, and the solutions in $\mathcal{P}$ that is dominated by the new solution would be excluded. POSS repeats the iteration T times, and it is proved to achieve the optimal approximation guarantee of $(1 - 1/e)$ with $E[T] \leq 2ek^2n$ expected running time (Qian et al., 2015).

Compared with POSS, POAL solves a more challenging problem, which has two essential differences: the number of objectives (or criteria in AL) and the constraint. Firstly, although POSS is in the form of bi-objective optimization, it supports only one criterion function, e.g., only $\mathcal{U}(\mathbf{x})$, while another criterion is that the subset size does not exceed b (i.e., $\sum_i s_i \leq b$). In contrast, POAL needs to solve multiple criterion functions. Secondly, POSS solves the problem with the constraint $|S| \leq k$, where as our setting involves fixed-size solutions $|S| = k$. Due to these differences, the theoretical guarantees in POSS cannot hold if we just adapt the POSS algorithm to our problem because: 1) Lemma 1 in POSS (Qian et al., 2015) is extracted from the proof of Theorem 3.2 in (Das & Kempe, 2011), which solves a Subset Selection problem as a single objective optimization problem with a greedy algorithm. If we simply add another objective, the theoretical

[4]General multi-objective evolutionary algorithms generate a population of solutions to update the Pareto set $\mathcal{P}$ at each iteration (Saxena et al., 2016), but our POAL only generates one solution at each iteration, which hardly improves $\mathcal{P}$. Thus, two successive Pareto sets $\mathcal{P}_{i-1}$ and $\mathcal{P}_i$ at the iterations $i - p_{inv}$ and i are compared in our POAL.

optimal approximation guarantee (Theorem 1 in (Qian et al., 2015)) cannot hold; 2) the POSS algorithm finds some optimal solution with the size $|S| < k$, which does not satisfy our solution requirement.

Considering the above issues, we devise our POAL to randomly generate candidate solutions with fixed batch sizes. As a result, this 1) controls all candidate solutions in $\mathcal{P}$ to have the same fixed size; 2) compared with POSS, Monte Carlo POAL does not depend on previous selections, which keeps the method free from initialization problems. Due to these differences in how candidate solutions are generated and the number of criteria, the theoretical bound on the number of iterations T needed for convergence of POSS (Qian et al., 2015) cannot be applied to MCPOAL. Thus, when should the iterations of our Monte-Carlo POAL terminate? (Qian et al., 2017) has proved that the Pareto set empirically converges much faster than $E[T]$ (see Fig. 2 in (Qian et al., 2017)). Based on (Qian et al., 2017) and (Saxena et al., 2016), we also propose an *early-stopping* technique. For the calculation of $E[T]$ in POAL, see Section A in Appendix.

4 Experiments

In the experiments we aim to: 1) evaluate the effectiveness of POAL on both classical ML and DL tasks under various scales of OOD data scenarios; 2) compare different multi-objective optimization strategies, i.e., weighted-sum, two-stage, and Pareto optimization; 3) observe how OOD data influence typical AL methods as discussed in Section 2.

4.1 Experimental Design

Datasets. For classical ML tasks, we use pre-processed data from LIBSVM (Chang & Lin, 2011):

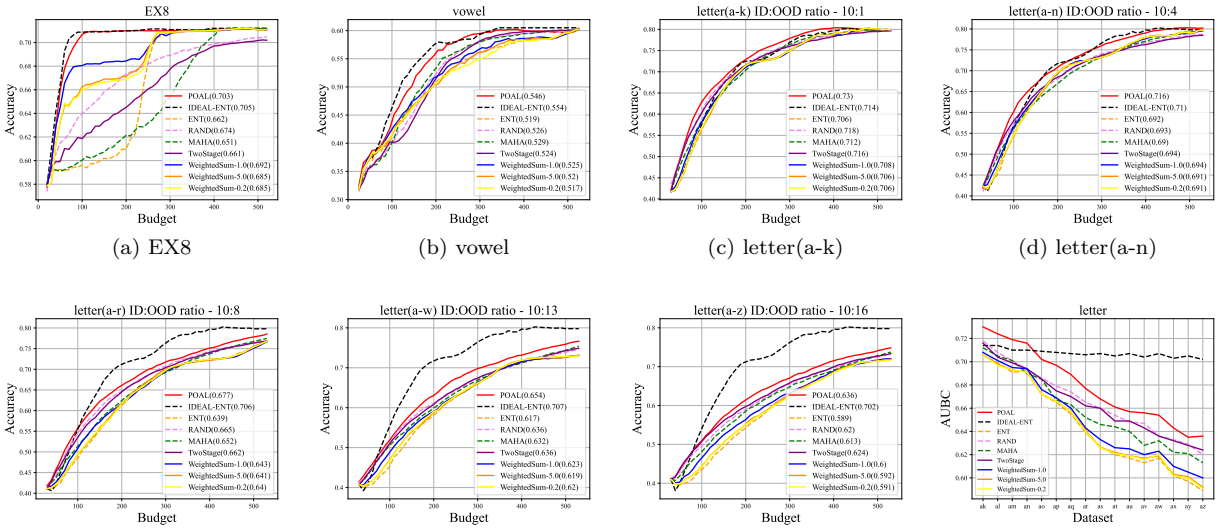
- Synthetic data: *EX8* uses *EX8a* as ID data and *EX8b* as OOD data (Ng, 2008).
- Real-life data: *Vowel* (Asuncion & Newman, 2007; Aggarwal & Sathe, 2015) has 11 classes, and we use 7 classes as ID data and the remaining for OOD data. *Letter* (Frey & Slate, 1991; Asuncion & Newman, 2007) has 26 classes, and we use 10 classes (*a*-*j*) as ID data and the remaining 16 classes (*k*-*z*) as OOD data. We also construct 16 datasets with increasing ID:OOD ratios, denoted as *letter(a-k)*, *letter(a-l)*,..., *letter(a-z)*.

For DL tasks, we adopt the following image datasets:

- *CIFAR10* (Krizhevsky et al., 2009) has 10 classes, and we construct two datasets: *CIFAR10-04* splits the classes with ID:OOD ratio of 6:4, and *CIFAR10-06* splits data with ratio as 4:6.
- *CIFAR100* (Krizhevsky et al., 2009) has 100 classes, and we construct *CIFAR100-04* and *CIFAR100-06* using the same ratios as *CIFAR10*.

Baselines. POAL helps existing AL methods select high informative ID data samples while preventing OOD data selection. In our experiments, we adopt ENT as our basic AL sampling strategy. We compare against three baseline methods considering OOD data: CCAL (Du et al., 2021), Deep Deterministic Uncertainty (DDU) (Mukhoti et al., 2021) with GMM density and SIMILAR (with FLVMI submodular function) (Kothawade et al., 2021). SIMILAR has many variants with different submodular functions. According to the experiments in (Kothawade et al., 2021), SIMILAR (FLCMI) perform better on OOD data scenarios, but it is memory-/time-consuming. We compared POAL with SIMILAR (FLCMI) in down-sampled *CIFAR10* dataset, followed the experimental settings in (Kothawade et al., 2021), as shown in Section C.2 in Appendix. We compare against five normal AL methods without OOD detection: ENT, BALD (Gal et al., 2017) and LPL (Yoo & Kweon, 2019) as uncertainty-based measures; k-Means as representative-based measure and BADGE (Ash et al., 2020) as combined strategy. From the aspect of multi-objective optimization, we compared MCPOAL against two alternative combination strategies: weighted sum optimization (WeightedSum-η) using weights $\eta = \{1.0, 5.0, 0.2\}$, where η refers to the ratio between the coefficients for the AL objective and OOD objective, and two-stage optimization (TwoStage) using threshold $\delta = \texttt{mean}(\mathbf{m})$. Additionally, we add random sampling (RAND) and Mahalanobis distance (MAHA) as baselines. Finally, we report the oracle results of ENT where only ID data is selected first (IDEAL-ENT), which serves as ideal performance of POAL.

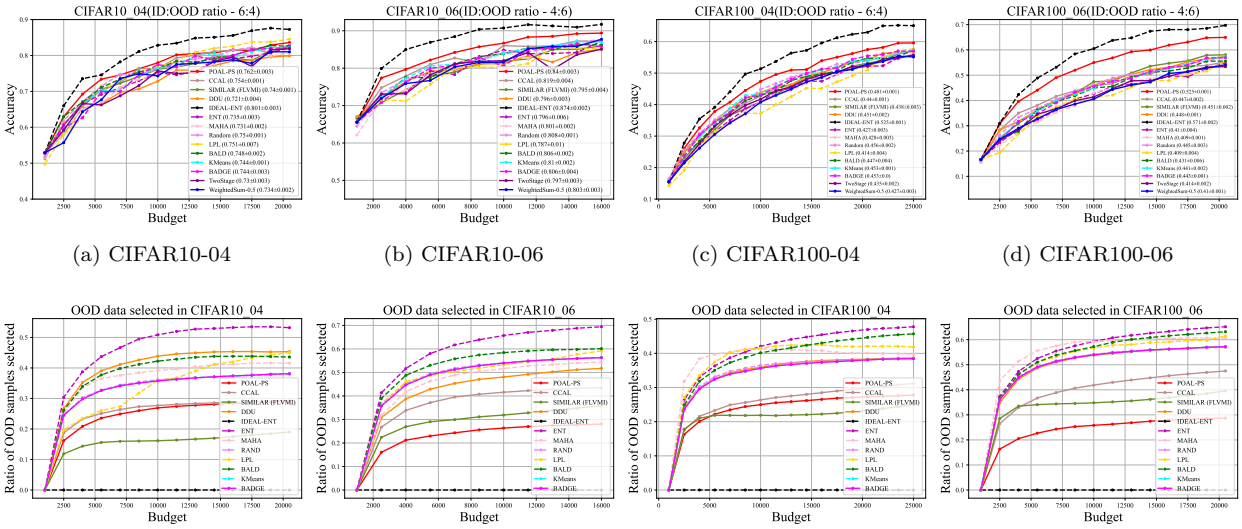
Implementation details. The training/test split of the datasets is fixed in the experiments, while the initial label set and the unlabeled set is randomly generated from the training set. Experiments are repeated 100 times for classical ML tasks and 3 times for DL tasks. We use the same basic classifier for each dataset, as shown in Table 7 in Appendix. To evaluate model performance, we measure accuracy and plot accuracy vs. budget curves to present the performance change with increasing labeled samples. To calculate average performance, we compute the area under the accuracy-budget curve (AUBC) (Zhan et al., 2021b), with higher values reflecting better overall performance under varying budgets.



(a) EX8 (b) vowel (c) letter(a-k) (d) letter(a-n)

(e) letter(a-r) (f) letter(a-w) (g) letter(a-z) (h) letter(AUBC)

Figure 3: (a)-(g) are accuracy vs. budget curves for classical ML tasks. The AUBC performances are shown in parentheses in the legend. To observe the effect of increasing ID:OOD ratio on *letter* datasets, we plot AUBC vs. dataset curves in (h).



(a) CIFAR10-04 (b) CIFAR10-06 (c) CIFAR100-04 (d) CIFAR100-06

(e) CIFAR10-04 OOD (f) CIFAR10-06 OOD (g) CIFAR100-04 OOD (h) CIFAR100-06 OOD

Figure 4: Results of POAL with **P**re-**S**election on DL datasets. (a-d) are accuracy vs. budget curves. We present the mean and standard deviation (in parentehsis) of AUBC. (e-h) plot the ratio of OOD samples selected.

4.2 Experimental Result Analysis

4.2.1 Overall Performance

We present the overall performance of classical ML and DL tasks in Fig. 3 and Fig. 4, respectively. We next analyze the overall performance from different aspects:

Performance on classical ML tasks. Considering each dataset (see data visualization in Appendix B.2), the ideal performance of POAL is shown on the synthetic dataset *EX8* (Fig. 3a), the ID data is non-linearly separable, and the OOD data is far from ID data. These properties make the calculation of $\mathcal{M}$ very accurate. Meanwhile, OOD and ID data close to the decision boundaries have high entropy scores. The curve of our POAL is the closest to IDEAL-ENT, which indicates that POAL selects the most informative ID data while excluding the OOD samples. For real-life classical ML datasets like *vowel* and *letter* (Fig. 3b-e), POAL also demonstrates its superiority. In *letter*, we fixed the ID data (letters *a*-*j*) and gradually increase the OOD data (letters *k*-*z*), and plot the result in Fig. 3h. We observe that except for IDEAL-ENT, all methods are influenced by increasing OOD data, e.g., the AUBC of RAND is 0.718 in Fig. 3c, 0.692 in Fig. 3d. Although our method is also influenced by increasing OOD data, it is still superior to all baselines.

The performance of POAL is influenced by the Mahalanobis distance calculation, we observe from Fig. 7, Fig. 8, and Fig. 9 that the capability of Mahalanobis distance is limited by the distinction between ID data distribution and OOD data distribution. *EX8* has distinct ID/OOD data distributions (see Fig. 7a), thus the Mahalanobis distance well distinguishes ID and OOD data, and reaches the optimal performance (Fig. 9a, MAHA has the same curve with IDEAL-ENT). However, for *vowel* and *letter* dataset, the ID/OOD data distributions are not as distinct as *EX8* (see Fig. 7b-r), thus the performance of MAHA is influenced. Furthermore, it affects the performance of our POAL. It makes our POAL behaves less perfectly on *vowel* and *letter* datasets than on the *EX8* dataset. Similar conclusions appear in (Ren et al., 2019).

Performance on DL tasks. POAL also works on large-scale data with DL tasks. Fig. 4 shows the accuracy-budget curves (along with the AUBC (acc) values) for *CIFAR10* and *CIFAR100* with ID:OOD ratios 6:4 and 4:6, together with the number of OOD samples selected during AL processes. Compared with normal AL methods, POAL outperforms both uncertainty-, representative/diversity-based and combined AL baselines like LPL, BALD, k-Means and BADGE on all tasks. As shown in Fig. 4e-h, the three AL under OOD data scenario methods, POAL, CCAL and SIMILAR effectively reduce the amount of OOD data that is selected. Both POAL and CCAL perform better than normal AL sampling schemes (i.e., ENT, LPL, BALD, k-Means, BADGE and RAND) on both *CIFAR10-04* and *CIFAR10-06* datasets.

POAL outperforms CCAL, SIMILAR (FLVMI) and DDU by large margins on more challenging situations (i.e., ID:OOD ratio is 4:6), POAL selects fewer OOD samples and achieves better AUBC performance (e.g., the AUBC value of POAL on *CIFAR100-06* is 0.525, while the highest baseline SIMILAR is 0.451). SIMILAR selects data samples close to labeled ID data and dissimilar to OOD data. It prevents selecting the OOD data, but the selected ID data may not be informative enough. CCAL utilized weighted-sum optimization to balance the informative and distinctive selections, and the trade-off parameter heavily affects its performance (also see Section 4.3 in (Du et al., 2021)). The idea of DDU (GMM) is similar to MAHA, the difference is DDU (GMM) does not use shared covariance as Eq. 4 while MAHA constructs class-conditional data distribution that follows the multivariate Gaussian distribution. Overall, DDU selects less OOD data than MAHA, especially in early stages. But both two methods select more OOD samples than POAL, CCAL and SIMILAR.

To observe the efficiency of each AL sampling strategy in our experiments of DL tasks, we summarized the overall performance across all models and datasets we adopted, including the mean and standard deviation (SD) of AUBC performance and running time. We record the running time from the start of the AL process to the output of the final basic learner. From Table 1, we can observe that our model outperforms all baselines (except for the ideal model – IDEAL-ENT) in terms of AUBC performance. For running time, POAL searches the near-optimal solution from the subset level, CCAL first extract semantic and distinctive features then calculates the pair-wise distance between large-scale unlabeled and labeled data. This results in the high computational costs of CCAL and POAL. The running time of POAL could be reduced by decreasing the

sliding window size s_w, although there may be some variations due to the Monte-Carlo sampling in MC POAL. Among POAL, CCAL and SIMILAR (FLVMI), which are designed specific to AL under OOD data scenarios, the running cost of our POAL is affordable.

Figure 4 also demonstrates the conflict between ID and OOD data in the unlabeled data pool within the current learning stage. The slope of the OOD-budget curve shows the degree of conflict between ID and OOD data, where a larger slope indicates a more severe conflict (more OOD data is sampled). Figure 4e&g have lower OOD ratios compared to Figure 4f&h, and we observed that standard AL measures, such as ENT, generally have higher slopes in situations with larger OOD ratios. This suggests that the degree of conflict is heavier in situations with larger OOD ratios. Additionally, we found that the slopes for all baselines, including POAL, are higher in previous learning stages and slow down in the middle and final stages, indicating that the degree of conflict decreases as the learning stage progresses (Figure 4e-h). Also note that at the beginning, all types of AL approaches are likely to encounter data insufficiency problems. Therefore, we first compared our AL approach with passive learning (RAND) and then compared our proposed method with other AL strategies. Our POAL consistently outperformed the baselines on DL tasks. In summary, while insufficient training data can be a problem in AL under OOD data scenarios, our approach effectively addresses this issue and performs well in comparison to other methods by selecting the near-optimal subsets in the current learning stages.

POAL vs. other multi-objective optimization strategies. We compared our Pareto optimization against weighted-sum, two-stage optimization, and each single objective, i.e., ENT and MAHA. Both ENT and MAHA are ineffective when used as a single selection strategy under OOD scenarios – in Fig 3a, ENT always selects OOD data with higher entropy until all OOD data are selected (*EX8* contains 210 OOD samples), and it performs even worse than RAND. MAHA prefers to select ID samples first (see Appendix Fig. 9), but the data samples with smaller Mahalanobis distance are also easily classified and far from the decision boundary, and thus not informative. POAL outperforms the other combination strategies, WeightedSum for different η values and TwoStage, on all tasks. This demonstrates that joint Pareto optimization better handles the two conflicting criteria. Considering WeightedSum optimization with different η values, we observe from Figure 3 that $\eta = 1.0$, i.e., treat each objective with the same importance, performs better than other η settings on various tasks. This observation is consistent with (Du et al., 2021; Ebert et al., 2012).

POAL vs. normal AL. We run experiments on normal AL methods, including uncertainty-based (ENT, LPL and BALD), diversity-based (k-Means) and combined methods (BADGE). Although LPL performs fairly well on standard *CIFAR10* and *CIFAR100* datasets (Yoo & Kweon, 2019; Zhan et al., 2022a), in OOD scenarios, OOD data have larger predicted loss values and thus influence the selection. Similar phenomena were observed with BALD and ENT. The performances of k-Means are close to RAND since both methods sampled ID/OOD data with the same ratio. Although k-Means will not select OOD samples first like uncertainty-based measures, it still failed to provide comparable performances. POAL performs well by selecting fewer OOD samples during the AL iterations, as shown in Fig. 4(e-h). BADGE performed similar to k-Means. The first stage of BADGE calculates the gradient embedding per sample, and use k-Means++ for clustering. These operations results in both high informative ID and OOD data are mixed together with high gradient values, therefore, BADGE failed to distinguish ID/OOD data. In DL tasks, we adopted pre-selection to deal with large-scale datasets. Although it makes the search space smaller, i.e., changes the global solutions to local solutions, POAL-PS still performs significantly better than baseline AL methods.

4.3 Visualization of the Subset Selection of POAL

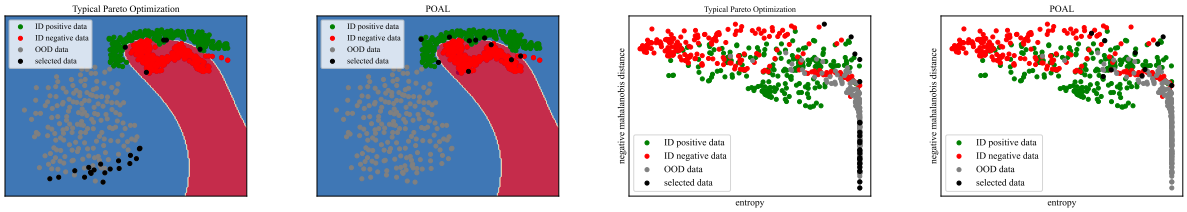
A direct way to validate the efficacy of POAL is to measure the difference between the selected subset by POAL and true Pareto set. However, on normal datasets, the searching space of the optimal subset selection is too large, as analysed in Section 3.4, it is a combination-level, $\mathcal{C}(M, b)$. Therefore, we use the subset selected by typical Pareto optimization (the operation is the same as the operation of the first round of pre-selection technique in Section 3.4) to represent the optimal subset selection instead. We visualize the subset selection on data and multi-objective space, as shown in Fig. 5. Using typical Pareto optimization will: i) select a subset with non-fixed subset size, and ii) possibly include many OOD data samples since the

Table 1: Overall comparison across all AL sampling strategies of all DL tasks, including **mean** and **standard deviation (SD)** of AUBC performance and running time (in seconds) across three repeated trials. The experimental environment deployed on DDU and MAHA are different from other methods. To maintain fairness, we use symbol "−" instead of running times.

	CIFAR10-04		*CIFAR10-06*	
	AUBC	Time	AUBC	Time
RAND	0.7500 (0.0014)	10805.3 (148.0)	0.8080 (0.0008)	5217.7 (348.0)
IDEAL-ENT	0.8007 (0.0029)	14578.0 (838.2)	0.8737 (0.0019)	9398.0 (696.4)
POAL ($s_w = 20$)	**0.7620** (0.0033)	62082.7 (7371.9)	0.8400 (0.0029)	34946.0 (4860.2)
POAL ($s_w = 10$)	0.7613 (0.0024)	56107.0 (3364.2)	**0.8403** (0.0029)	47362.0 (188.7)
CCAL	0.7543 (0.0009)	76129.3 (5853.7)	0.8190 (0.0037)	35715.3 (328.5)
SIMILAR (FLVMI)	0.7397 (0.0012)	13942.7 (116.2)	0.7947 (0.0037)	8550.0 (231.3)
DDU (GMM)	0.7207 (0.0038)	− −	0.7957 (0.0025)	− −
ENT	0.7350 (0.0029)	10155.3 (872.3)	0.7960 (0.0057)	5485.7 (655.4)
MAHA	0.7313 (0.0021)	− −	0.8013 (0.0021)	− −
LPL	0.7510 (0.0071)	12384.7 (1555.9)	0.7873 (0.0103)	3783.3 (326.9)
BADGE	0.7437 (0.0029)	41646.0 (12723.5)	0.8063 (0.0037)	17539.3 (628.0)
KMeans	0.7440 (0.0014)	18785.7 (2571.3)	0.8100 (0.0022)	8627.3 (407.2)
BALD	0.7480 (0.0022)	10478.3 (679.7)	0.8060 (0.0022)	4158.3 (431.1)
TwoStage	0.7300 (0.0029)	8342.3 (1059.4)	0.7970 (0.0028)	5524.7 (821.3)
WeightedSum-1.0	0.7337 (0.0019)	9489.7 (454.5)	0.8033 (0.0033)	2006.3 (13.8)
WeightedSum-5.0	0.7327 (0.0029)	11892.3 (1733.9)	0.8060 (0.0045)	7998.3 (81.6)
WeightedSum-0.2	0.7340 (0.0045)	8249.3 (2019.2)	0.8063 (0.0012)	5751.0 (795.3)
	CIFAR100-04		*CIFAR100-06*	
	AUBC	Time	AUBC	Time
RAND	0.4560 (0.0016)	11563.3 (985.5)	0.4453 (0.0026)	11075.3 (1198.2)
IDEAL-ENT	0.5250 (0.0008)	17736.7 (1694.7)	0.5707 (0.0017)	19098.0 (1458.8)
POAL ($s_w = 20$)	**0.4807** (0.0009)	66880.7 (1583.5)	0.5253 (0.0005)	62762.0 (4277.4)
POAL ($s_w = 10$)	0.4797 (0.0017)	4932.3 (1891.2)	**0.5270** (0.0008)	55955.3 (5162.4)
CCAL	0.4400 (0.0008)	21253.0 (2171.9)	0.4467 (0.0024)	65303.0 (1676.6)
SIMILAR (FLVMI)	0.4377 (0.0026)	20732.7 (662.1)	0.4510 (0.0024)	13137.0 (396.7)
DDU (GMM)	0.4510 (0.0022)	− −	0.4483 (0.0012)	− −
ENT	0.4267 (0.0034)	11202.3 (2484.7)	0.4100 (0.0036)	10940.0 (1318.4)
MAHA	0.4283 (0.0026)	− −	0.4090 (0.0008)	− −
LPL	0.4140 (0.0037)	17797.0 (4093.8)	0.4087 (0.0042)	5633.0 (252.3)
BADGE	0.4530 (0.0000)	58877.7 (13876.3)	0.4430 (0.0008)	58046.3 (17296.9)
KMeans	0.4527 (0.0005)	34944.3 (4121.3)	0.4413 (0.0021)	17465.0 (3570.3)
BALD	0.4467 (0.0040)	14699.0 (2675.9)	0.4313 (0.0061)	4574.3 (140.5)
TwoStage	0.4347 (0.0021)	11484.3 (4360.1)	0.4137 (0.0017)	3234.3 (91.1)
WeightedSum-1.0	0.4267 (0.0025)	12722.0 (4164.8)	0.4103 (0.0005)	9455.3 (468.5)
WeightedSum-5.0	0.4287 (0.0039)	14822.3 (6249.7)	0.4073 (0.0050)	11309.3 (1303.4)
WeightedSum-0.2	0.4290 (0.0022)	11736.7 (4394.3)	0.4097 (0.0033)	8865.7 (1230.1)

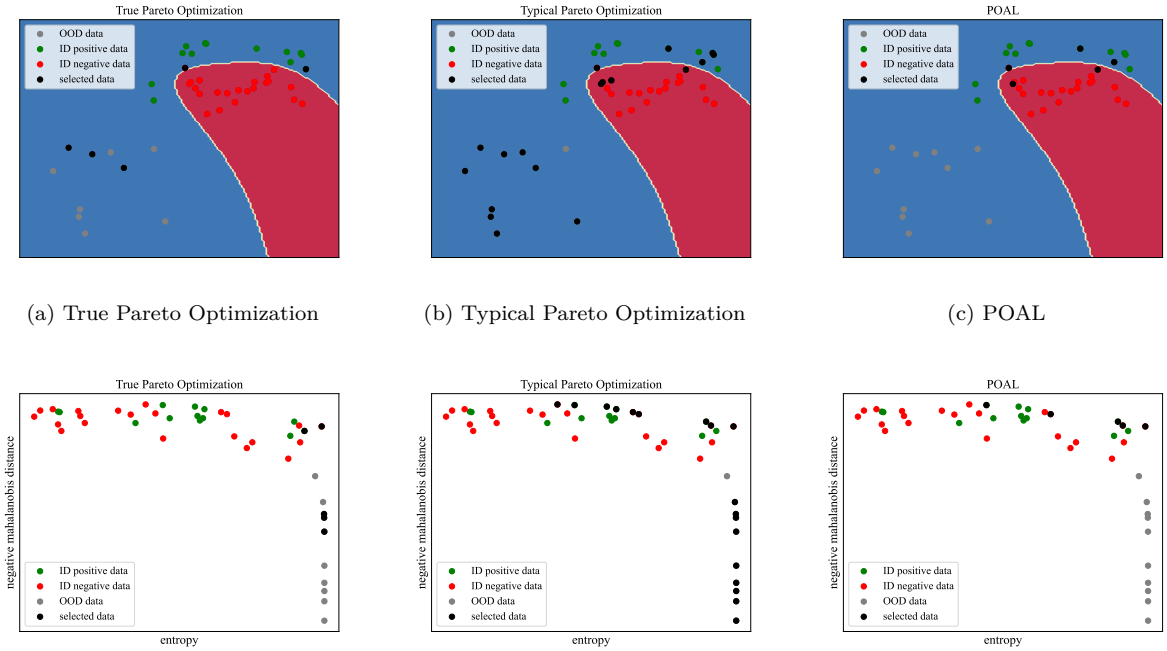
entropy score of OOD data samples are relatively large. In contrast, POAL selects more informative ID data and less OOD samples based on current basic learners in Fig. 5.

We conduct a toy experiments to show whether the subset selection generated by POAL is similar to the output generated by True Pareto Optimization for subset selection, based on exhaustive search, as shown in Figures 6 and 1c. We down-sampled the unlabeled data pool to size of 50 and set the batch size as 5. The resulting search space is of size $\mathcal{C}(50, 5) = 2,118,760$, which is acceptable. We traverse the whole search space and use the same Final-selection method as in POAL to generate a single final solution. As shown in Fig. 1c, we observe that the candidate Pareto set $\mathcal{P}$ generated by POAL is close to the selection by True Pareto Optimization. In v, the solutions generated by POAL are located on Pareto Curve (or very close to the Pareto Curve), in which Pareto Curve is generated by the True Pareto Optimization. Observing Fig. 6a-f, the subset selection generate by POAL is better. This is because True Pareto Optimization finally selects the subset with higher entropy and lower ID score, which might select OOD samples.



(a) Typical Pareto Optimization (b) POAL (c) Typical Pareto Optimization (d) POAL

Figure 5: The subset selection generated by POAL and Typical Pareto Optimization. The selection results are presented on data space and multi-objective space respectively. The experimental settings are the same as Fig. 1.



(a) True Pareto Optimization (b) Typical Pareto Optimization (c) POAL

(d) True Pareto Optimization (e) Typical Pareto Optimization (f) POAL

Figure 6: The subset selection generated by POAL, True Pareto Optimization (via exhaustive search) and Typical Pareto Optimization on down-sampled *EX8* dataset. The selection results are presented on data space and multi-objective space respectively. a-f are based on single data point level. The x-axis and y-axis are $\mathcal{U}$ and $\mathcal{M}$ respectively.

4.4 Sensitivity Analysis

In our work, we have a hyper-parameter s_w (sliding window size) to control the early stopping condition. Large s_w applies to a more strict early stop condition and vice versa. If there is no significant change within $s_w \times p_{\text{inv}}$ iterations, then we can stop early. In our experiments, we set $s_w = 20$. In this section, we conduct an ablation study to see if less strict or more strict conditions influence the final performance, as shown in Table 1, 3rd and 4th rows of each table. We found that $s_w = 10$ (less strict) does not affect the model performance (there is no significant difference of AUBC performance between $s_w = 20$ and $s_w = 10$). Less strict early stopping also usually requires less running time, e.g., in *CIFAR100-04* dataset, we have 66,880.7 seconds average running time with $s_w = 20$ and 4,932.3 seconds average running time with $s_w = 10$.

Besides s_w, we have another hyper-parameter s_m, which controls the minimum pre-selected subset size. s_m is designed to adapt to various computational resource constraints, but not for controlling the AL/OOD

trade-off. Larger s_m can be used when more computational resources/time are available, while smaller s_m can save computational cost but may lead to slightly sub-optimal solutions. In our experiments, we set $s_m = 6b$ based on our computational resource situation, which works well in practice.

Although the numbers of hyper-parameters of POAL are similar to the weighted-sum optimization, their purposes are fundamentally different. In the weighted-sum optimization, hyper-parameters relate to the weighting of the two objectives, and thus significantly influence the result. In contrast, the two hyper-parameters of POAL are about the efficient approximation (concerning computational resources), which mainly affects the initialization and stopping criteria, possibly resulting in sub-optimal approximate solutions. Note that we do not need to tune these parameters for various tasks.

5 Conclusion

In this paper, we proposed an effective and flexible Monte Carlo POAL framework for improving AL when encountering OOD data in the unlabeled data pool. Our POAL framework i) incorporates various AL sampling strategies; ii) handles large-scale datasets; iii) supports batch-mode AL with fixed batch size; iv) does not require tuning a trade-off hyper-parameter; and v) works well on both classical ML and DL tasks. Future work could improve the OOD detection criterion by introducing OOD-specific feature representations and other techniques for better distinguishing ID/OOD distributions.

Author Contributions

The first two authors, Xueying Zhan and Zeyu Dai, are equally contributed this paper. Xueying Zhan contributed the model implementation, draft manuscript preparation, analysis and interpretation of results and study conception and design parts. Zeyu Dai contributed the study conception and design, analysis and interpretation of results and draft manuscript preparation parts.

Acknowledgments

The work was supported by a Strategic Research Grant from City University of Hong Kong (Project No. 7005840).

References

Charu C Aggarwal and Saket Sathe. Theoretical foundations and algorithms for outlier ensembles. *Acm sigkdd explorations newsletter*, 17(1):24–47, 2015.

Shabbir Ahmed, Mohit Tawarmalani, and Nikolaos V Sahinidis. A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming*, 100(2):355–377, 2004.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9368–9377, 2018.

Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 49–56, 2009.

Viktor Bindewald, Fabian Dunke, and Stefan Nickel. Modeling multi-stage decision making under incomplete and uncertain information. Technical report, Technical report, 2020.

Erdem Bıyık, Kenneth Wang, Nima Anari, and Dorsa Sadigh. Batch active learning using determinantal point processes. *arXiv preprint arXiv:1906.07975*, 2019.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

PM Chaudhari, RV Dharaskar, and VM Thakare. Computing the most significant solution from pareto front obtained in multi-objective evolutionary. *International Journal of Advanced Computer Science and Applications*, 1(4), 2010.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Yufei Cui, Wuguannan Yao, Qiao Li, Antoni B Chan, and Chun Jason Xue. Accelerating monte carlo bayesian prediction via approximating predictive uncertainty over the simplex. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Abhimanyu Das and David Kempe. Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 1057–1064, 2011.

Geoff Davis, Stephane Mallat, and Marco Avellaneda. Adaptive greedy approximations. *Constructive approximation*, 13(1):57–98, 1997.

Antoine de Mathelin, François Deheeger, Mathilde Mougeot, and Nicolas Vayatis. Discrepancy-based active learning for domain adaptation. In *International Conference on Learning Representations*, 2021.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Liat Ein Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. Active learning for bert: an empirical study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7949–7962, 2020.

Pan Du, Suyun Zhao, Hui Chen, Shuwen Chai, Hong Chen, and Cuiping Li. Contrastive coding for active learning under class distribution mismatch. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8927–8936, 2021.

Long Duong, Hadi Afshar, Dominique Estival, Glen Pink, Philip R Cohen, and Mark Johnson. Active learning for deep semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 43–48, 2018.

Sandra Ebert, Mario Fritz, and Bernt Schiele. Ralf: A reinforced active learning formulation for object class recognition. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3626–3633. IEEE, 2012.

Sebastian Farquhar, Yarin Gal, and Tom Rainforth. On statistical bias in active learning: How and when to fix it. *arXiv preprint arXiv:2101.11665*, 2021.

Richard Fredlund, Richard M Everson, and Jonathan E Fieldsend. A bayesian framework for active learning. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2010.

Peter W Frey and David J Slate. Letter recognition using holland-style adaptive classifiers. *Machine learning*, 6(2):161–182, 1991.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pp. 1183–1192. PMLR, 2017.

Ravi Ganti and Alexander Gray. Upal: Unbiased pool based active learning. In *Artificial Intelligence and Statistics*, pp. 422–431. PMLR, 2012.

Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19, 2006.

Willem K Klein Haneveld and Maarten H van der Vlerk. Stochastic integer programming: General models and algorithms. *Annals of operations research*, 85:39, 1999.

Elmar Haussmann, Michele Fenzi, Kashyap Chitta, Jan Ivanecky, Hanson Xu, Donna Roy, Akshita Mittel, Nicolas Koumchatzky, Clement Farabet, and Jose M Alvarez. Scalable active learning for object detection. In *2020 IEEE intelligent vehicles symposium (iv)*, pp. 1430–1435. IEEE, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Henrik Imberg, Johan Jonasson, and Marina Axelson-Fisk. Optimal sampling in unbiased active learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 559–569. PMLR, 2020.

Siddharth Karamcheti, Ranjay Krishna, Li Fei-Fei, and Christopher D Manning. Mind your outliers! investigating the negative impact of outliers on active learning for visual question answering. *arXiv preprint arXiv:2107.02331*, 2021.

Andreas Kirsch, Sebastian Farquhar, Parmida Atighehchian, Andrew Jesson, Frederic Branchaud-Charron, and Yarin Gal. Stochastic batch acquisition for deep active learning. *arXiv preprint arXiv:2106.12059*, 2021.

Suraj Kothawade, Nathan Beck, Krishnateja Killamsetty, and Rishabh Iyer. Similar: Submodular information measures based active learning in realistic scenarios. *Advances in Neural Information Processing Systems*, 34, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine Learning, Proceedings of the Eleventh International Conference, Rutgers University, New Brunswick, NJ, USA, July 10-13, 1994*, pp. 148–156. Morgan Kaufmann, 1994.

Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Shengcai Liu, Ning Lu, Cheng Chen, Chao Qian, and Ke Tang. Hydratext: Multi-objective optimization for adversarial textual attack. *arXiv preprint arXiv:2111.01528*, 2021.

Rafid Mahmood, Sanja Fidler, and Marc T Law. Low budget active learning via wasserstein distance: An integer programming approach. *arXiv preprint arXiv:2106.02968*, 2021.

R Timothy Marler and Jasbir S Arora. The weighted sum method for multi-objective optimization: new insights. *Structural and multidisciplinary optimization*, 41(6):853–862, 2010.

Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip HS Torr, and Yarin Gal. Deep deterministic uncertainty: A simple baseline. *arXiv e-prints*, pp. arXiv–2102, 2021.

Andrew Ng. Stanford cs229 - machine learning - andrew ng. 2008.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Chao Qian, Yang Yu, and Zhi-Hua Zhou. Subset selection by pareto optimization. *Advances in neural information processing systems*, 28, 2015.

Chao Qian, Jing-Cheng Shi, Yang Yu, Ke Tang, and Zhi-Hua Zhou. Optimizing ratio of monotone set functions. In *IJCAI*, pp. 2606–2612, 2017.

Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. *Advances in Neural Information Processing Systems*, 32, 2019.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM Computing Surveys (CSUR)*, 54(9):1–40, 2021.

Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009.

Ahti Salo, Juho Andelmin, and Fabricio Oliveira. Decision programming for mixed-integer multi-stage optimization under uncertainty. *European Journal of Operational Research*, 299(2):550–565, 2022.

Christoph Sawade, Niels Landwehr, Steffen Bickel, and Tobias Scheffer. Active risk estimation. In *ICML*, 2010.

Dhish Kumar Saxena, A. Sinha, João A. Duro, and Q. Zhang. Entropy-based termination criterion for multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20(4):485–498, 2016.

Panos Seferlis and Michael C Georgiadis. *The integration of process design and control*. Elsevier, 2004.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Burr Settles. Active learning literature survey. 2009.

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 287–294, 1992.

Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew Lim Tan. Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain*, pp. 589–596. ACL, 2004.

Changjian Shui, Fan Zhou, Christian Gagné, and Boyu Wang. Deep active learning: Unified and principled method for query and training. In *International Conference on Artificial Intelligence and Statistics*, pp. 1308–1318. PMLR, 2020.

Sagar Vaze, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Open-set recognition: A good closed-set classifier is all you need. *arXiv preprint arXiv:2110.06207*, 2021.

Dan Wang and Yi Shang. A new active labeling method for deep learning. In *2014 International joint conference on neural networks (IJCNN)*, pp. 112–119. IEEE, 2014.

Jean-Paul Watson, Harvey J Greenberg, and William E Hart. A multiple-objective analysis of sensor placement optimization in water networks. In *Critical transitions in water and environmental resources management*, pp. 1–10. 2004.

Hongxin Wei, Lue Tao, Renchunzi Xie, and Bo An. Open-set label noise can improve robustness against inherent label noise. *Advances in Neural Information Processing Systems*, 34:7978–7992, 2021.

Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

Changchang Yin, Buyue Qian, Shilei Cao, Xiaoyu Li, Jishang Wei, Qinghua Zheng, and Ian Davidson. Deep similarity-based batch mode active learning with exploration-exploitation. In *2017 IEEE International Conference on Data Mining (ICDM)*, pp. 575–584. IEEE, 2017.

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

Xueying Zhan, Qing Li, and Antoni B Chan. Multiple-criteria based active learning with fixed-size determinantal point processes. *arXiv preprint arXiv:2107.01622*, 2021a.

Xueying Zhan, Huan Liu, Qing Li, and Antoni B. Chan. A comparative survey: Benchmarking for pool-based active learning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 4679–4686, 2021b.

Xueying Zhan, Qingzhong Wang, Kuan-hao Huang, Haoyi Xiong, Dejing Dou, and Antoni B Chan. A comparative survey of deep active learning. *arXiv preprint arXiv:2203.13450*, 2022a.

Xueying Zhan, Yaowei Wang, and Antoni B Chan. Asymptotic optimality for active learning processes. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022b.

Eric Zhao, Anqi Liu, Animashree Anandkumar, and Yisong Yue. Active learning under label shift. In *International Conference on Artificial Intelligence and Statistics*, pp. 3412–3420. PMLR, 2021.

Yu Zhao, Zhenhui Shi, Jingyang Zhang, Dong Chen, and Lixu Gu. A novel active learning framework for classification: Using weighted rank aggregation to achieve multiple query criteria. *Pattern Recognition*, 93: 581–602, 2019.

Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and evolutionary computation*, 1(1):32–49, 2011.

Appendix

A Computational Complexity Analysis of POAL

A.1 Monte-Carlo POAL

Our proposed Monte-Carlo POAL for fixed-size subset selection used unordered sampling with replacement. Denote the ground set $\mathcal{A}$ with size N. At each iteration, B (batch size, $B < N$) samples are randomly selected. So the search space contains $S = C_N^B$ combinations. The worst case of POAL is to find all Pareto optimal solutions $\mathcal{P} = \{s_1, s_2, \cdots\}$ out of S combinations (each solution is a subset of B distinct samples out of N samples). Let $|\mathcal{P}| = M$ ($M \leq S$). Hence, the computational complexity is the expected number of iterations that all of the M Pareto optimal solutions have been selected at least once, which is calculated below:

At each iteration, there is a probability $\frac{S-1}{S}$ that one particular solution s_i is not selected. So after k iterations, there is probability $(\frac{S-1}{S})^k$ that solution s_i has not been selected. Thus, the probability that solution s_i has been selected at least once after k iterations is $1 - (\frac{S-1}{S})^k$. For all M solutions, the probability that they have all been selected at least once after k iterations is $(1 - (\frac{S-1}{S})^k)^M$. To obtain the probability that these M solutions have been selected at least once **exactly** after k iterations (i.e., not before k iterations), we need to subtract the probability that it is completed after $k-1$ iterations: $(1 - (\frac{S-1}{S})^k)^M - (1 - (\frac{S-1}{S})^{k-1})^M$. Hence, the expected number of iterations is:

$$
\begin{aligned}
E[T] &= \sum_{k=0}^{\infty} k \cdot ((1 - (\frac{S-1}{S})^k)^M - (1 - (\frac{S-1}{S})^{k-1})^M) \\
&= \sum_{k=0}^{\infty} 1 - (1 - (\frac{S-1}{S})^k)^M.
\end{aligned}
$$

For small datasets, the size is tolerable, so the ground set $\mathcal{A}$ is just the unlabeled dataset $\mathcal{D}_u$ with size n, and the search space size $S = C_n^B$. In practice, we applied an early-stopping strategy, which can obtain a good enough solution with much fewer iterations.

A.2 Pre-selection for large-scale datasets

We designed a pre-selection technique for large-scale datasets to reduce the search space of POAL. As introduced in the last paragraph in Section 3.4, we iteratively select Pareto optimal samples from the unlabeled data. The worst case is that all unlabeled samples are Pareto optimal samples, so the number of comparisons is $0 + 1 + 2 + \cdots + (n-1) = \frac{n^2}{2} - n$. The computational complexity of the pre-selection is $O(n^2)$.

We pre-selected $s_m = \alpha B$ samples out of n unlabeled samples (α is set according to the computing resources and time budget, we set $\alpha = 6$ in our experiment) for the subsequent Monte-Carlo POAL. Therefore, $N = \alpha B \ll n$, and the search space size S is reduced from C_n^B to $C_{\alpha B}^B$. Also, with an early-stopping strategy, our POAL can achieve a good enough solution with an acceptable number of iterations in practice.

B Experimental Settings

B.1 Datasets

We summarize the datasets we adopted in our experiments in Table 2, including how to split the datasets (initial labeled data, unlabeled data pool, and test set), dataset information (number of categories, number of feature dimensions), and task-concerned information (maximum budget, batch size, numbers of repeated trials and basic learners adopted in each task). Primarily, we recorded the ID : OOD ratio in each task. We down-sampled the *letter* dataset and controlled that each category only has 50 data samples in the training set.

Table 2: Datasets used in the experiments.

Dataset	# of ID classes	# of feature dimension	# of initial labeled set	# of unlabeled pool	# of test set	# of Maximum Budget	ID : OOD Ratio	batch size b	# of repeat trials	basic learner
EX8	2	2	20	650	306	500	46 : 21	10	100	GPC
vowel	7	10	25	503	294	500	336 : 192	10	100	GPC
letter(a-k)	10	16	30	520	500	550	10 : 1	10	100	LR
letter(a-l)	10	16	30	570	500	550	10 : 2	10	100	LR
letter(a-m)	10	16	30	620	500	550	10 : 3	10	100	LR
letter(a-n)	10	16	30	670	500	550	10 : 4	10	100	LR
letter(a-o)	10	16	30	720	500	550	10 : 5	10	100	LR
letter(a-p)	10	16	30	770	500	550	10 : 6	10	100	LR
letter(a-q)	10	16	30	720	500	550	10 : 7	10	100	LR
letter(a-r)	10	16	30	870	500	550	10 : 8	10	100	LR
letter(a-s)	10	16	30	920	500	550	10 : 9	10	100	LR
letter(a-t)	10	16	30	970	500	550	10 : 10	10	100	LR
letter(a-u)	10	16	30	1020	500	550	10 : 11	10	100	LR
letter(a-v)	10	16	30	1070	500	550	10 : 12	10	100	LR
letter(a-w)	10	16	30	1120	500	550	10 : 13	10	100	LR
letter(a-x)	10	16	30	1170	500	550	10 : 14	10	100	LR
letter(a-y)	10	16	30	1220	500	550	10 : 15	10	100	LR
letter(a-z)	10	16	30	1270	500	550	10 : 16	10	100	LR
CIFAR10-04	6	$32 \times 32 \times 3$	1000	49000	6000	20000	6 : 4	500	3	ResNet18
CIFAR10-06	4	$32 \times 32 \times 3$	1000	49000	4000	15000	4 : 6	500	3	ResNet18
CIFAR100-04	60	$32 \times 32 \times 3$	1000	49000	6000	25000	6 : 4	500	3	ResNet18
CIFAR100-06	40	$32 \times 32 \times 3$	1000	49000	4000	20000	4 : 10	500	3	ResNet18
Down-sampled *CIFAR10*	8	$32 \times 32 \times 3$	1600	14000	8000	2250	8:2	125	3	ResNet18

B.2 Visualization of Datasets

B.2.1 Classical ML tasks.

We visualize the datasets of classical ML tasks using t-Distributed Stochastic Neighbor Embedding (t-SNE)[5], as shown in Fig. 7. We split ID/OOD data by setting OOD data samples as semitransparent grey dots to better observe the distinction between the ID and OOD data distributions. On *EX8* dataset, there is a large distinction between ID and OOD samples, thus the calculation of Mahalanobis distance score would be very accurate, see Fig. 9a as example. In contrast, if the gap between ID and OOD data is small, like *letter*(a-z), the ability of Mahalanobis distance score for distinguishing ID and OOD data will be limited, as shown in Fig. 9r.

B.2.2 DL tasks.

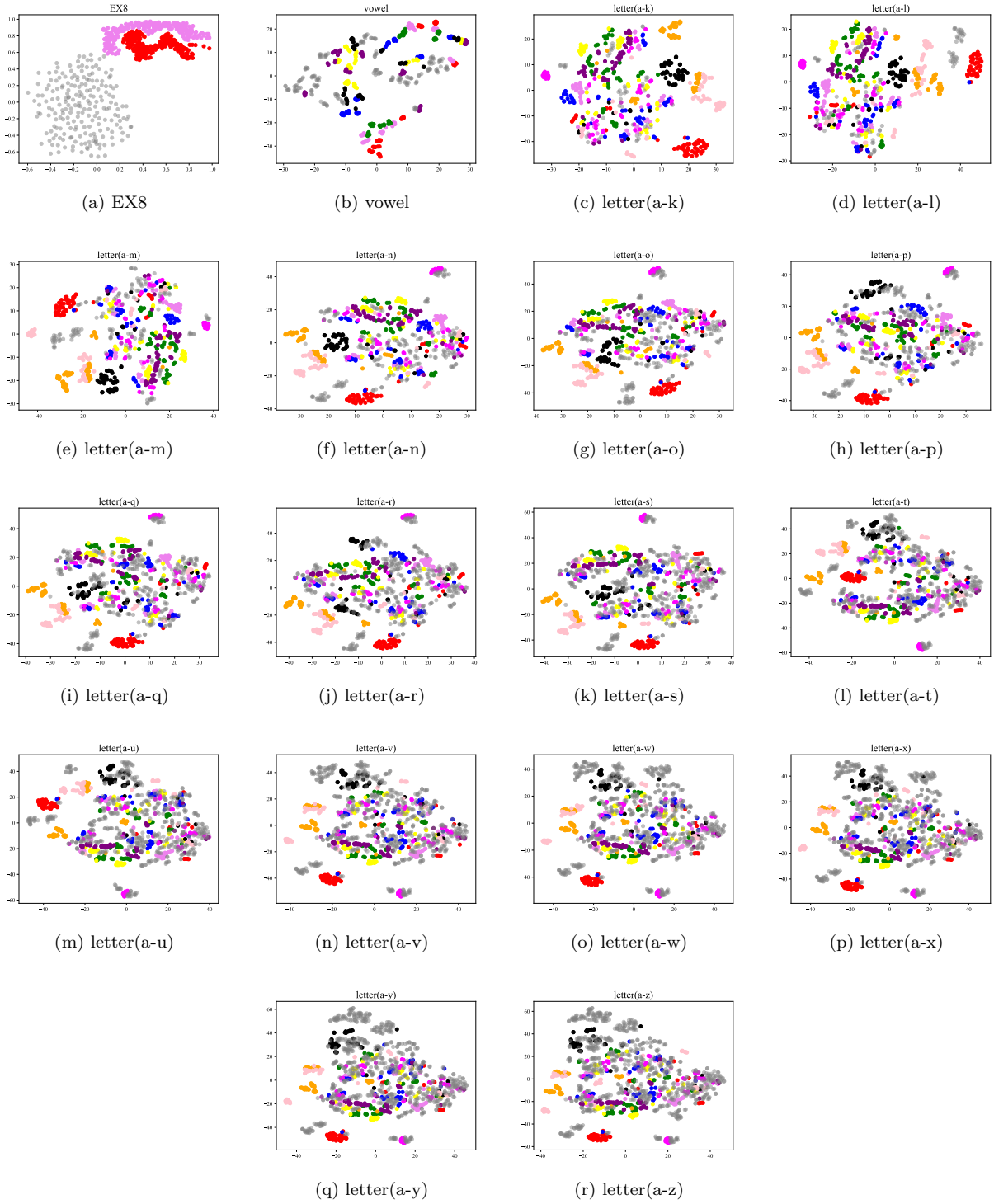
We also visualize the discrepancy using MMD in DL tasks since the raw features of DL tasks are hard to visualize, like Fig. 7. We calculate the MMD distance within the ID data and the MMD distance between the ID and OOD data. To extract the feature representations in DL tasks, we utilized a ResNet50 which is pre-trained on ImageNet dataset[6]. After extracting the features (the output of the penultimate layer of the pre-trained ResNet50) of ID and OOD samples, we next calculate the MMD scores, which is the same as the MMD settings in our early-stopping technique. Calculating the MMD score between these ID and OOD data samples is memory-consuming (CIFAR10 and CIFAR100 have $50,000$ training sets). To calculate the ID-ID MMD score, we randomly sampled $1,000 \times 2$ instances in ID data and calculated the MMD score. We repeated this operation 100 times and took the average score. For calculating the ID-OOD score, we perform the same operation.

The MMD scores of *CIFAR10-04*, *CIFAR10-06*, *CIFAR100-04* and *CIFAR100-06* are shown in Table 3. The distance between ID and OOD data are $1.5x$/ $2.5x$ larger compared with the in-between distance of ID data, which indicates that using the Mahalanobis distance for distinguishing ID and OOD data is feasible on *CIFAR10* and *CIFAR100* datasets. Note that we did not calculate the MMD scores of ID-ID and ID-OOD during each stage of the AL process, since calculating the MMD scores on large-scale data with high-dimensional feature representations is time- and memory-consuming.

B.3 Baselines and Implementations

We introduce the baselines and the implementation details in this section.

[5] https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
[6] https://pytorch.org/vision/main/models/generated/torchvision.models.resnet50.html



(a) EX8 (b) vowel (c) letter(a-k) (d) letter(a-l)

(e) letter(a-m) (f) letter(a-n) (g) letter(a-o) (h) letter(a-p)

(i) letter(a-q) (j) letter(a-r) (k) letter(a-s) (l) letter(a-t)

(m) letter(a-u) (n) letter(a-v) (o) letter(a-w) (p) letter(a-x)

(q) letter(a-y) (r) letter(a-z)

Figure 7: Visualization of data sets in classical ML tasks by t-SNE. We set the semitransparent grey dots to represent OOD data samples, the remaining colorful dots are ID data samples.

For the basic learner/classifier, we adopted Gaussian Process Classifier (GPC), Logistic Regression (LR), and ResNet18 in our experiments. We did not choose GPC in the *letter* dataset since the accuracy-budget curves based on GPC are not monotonically increasing. The requirement of selecting a basic classifier is: that the

Table 3: MMD scores of ID-ID and ID-OOD data in DL tasks.

	ID-ID	ID-OOD
CIFAR10-04	0.00608	0.04645
CIFAR10-06	0.00596	0.03565
CIFAR100-04	0.00603	0.01305
CIFAR100-06	0.00604	0.01659

basic classifier can provide soft outputs, that is, predictive class probabilities to calculate the uncertainty of each unlabeled sample, e.g., entropy. We use the implementation of the GPC with RBF kernel[7] and LR[8] of scikit-learn library (Pedregosa et al., 2011) with default settings. For ResNet18, we employed ResNet18 (He et al., 2016) based on PyTorch with Adam optimizer (learning rate: $1e-3$) as the basic learner in DL tasks. In *CIFAR10* and *CIFAR100* tasks, we set the number of training epochs as 30, the kernel size of the first convolutional layer in ResNet18 is 3×3 (consistent with PyTorch CIFAR implementation[9]). Input pre-processing steps include random crop (pad $= 4$), random horizontal flip ($p = 0.5$) and normalization.

For the implementation of the classical ML baselines, we have introduced it in the main paper.

For the baselines in DL tasks, we use the implementation of DeepAL+[10](Zhan et al., 2022a). We provide simple introductions of BALD, LPL and BADGE as follows:

- Bayesian Active Learning by Disagreements (BALD) (Houlsby et al., 2011; Gal et al., 2017): it chooses the data points that are expected to maximize the information gained from the model parameters, i.e., the mutual information between predictions and model posterior.
- Loss Prediction Loss (LPL) (Yoo & Kweon, 2019): it is a loss prediction strategy by attaching a small parametric module that is trained to predict the loss of unlabeled inputs concerning the target model by minimizing the loss prediction loss between predicted loss and target loss. LPL picks the top b data samples with the highest predicted loss.
- Batch Active learning by Diverse Gradient Embeddings (BADGE) (Ash et al., 2020): it first measures uncertainty as the gradient magnitude for the parameters in the output layer in the first stage; it then clusters the samples by k-Means++ in the second stage.

For CCAL, We utilize the open-source code implementation of CCAL[11]. We train SimCLR (Chen et al., 2020), the semantic/distinctive feature extractor, which is provided by CCAL's source code. We train the two feature extractors with 700 epochs and a batch size of 32 on a single V100 GPU.

We run all our experiments on a single Tesla V100-SXM2 GPU with 16GB memory except for running SIMILAR (FLCMI) related experiments. Since SIMILAR (FLCMI) needs much memory. We run the experiments (SIMILAR on down-sampled *CIFAR10*) on another workstation with Tesla V100-SXM2 GPU with 94GB memory in total.

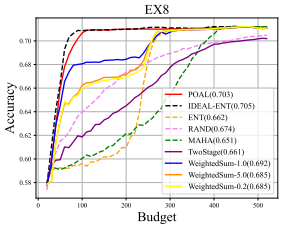
C Experiments

This section is an extension of Section 4 in the main paper. We provide additional experimental results analysis and more experiments that do not appear in the main paper.

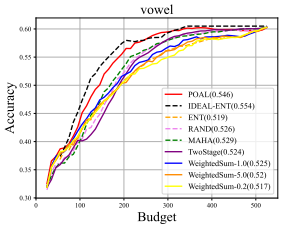
C.1 Complete Experiments of POAL on Classical ML Tasks

We present the complete accuracy vs. budget curves and numbers of OOD samples selected vs. budget curves during AL processes in Fig. 8 and Fig. 9 respectively.

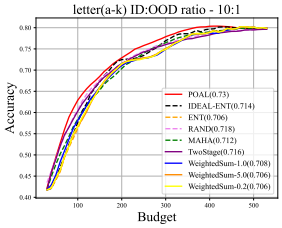
[7] https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.GaussianProcessClassifier.html
[8] https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
[9] https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py
[10] https://github.com/SineZHAN/deepALplus
[11] https://github.com/RUC-DWBI-ML/CCAL/



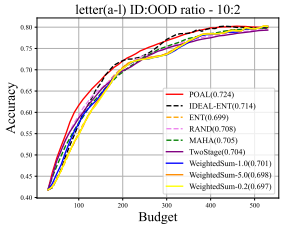
(a) EX8



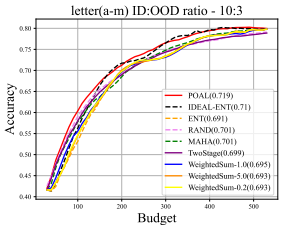
(b) vowel



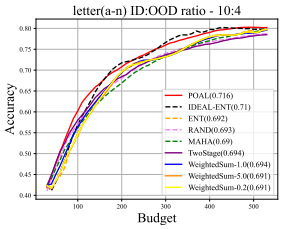
(c) letter(a-k)



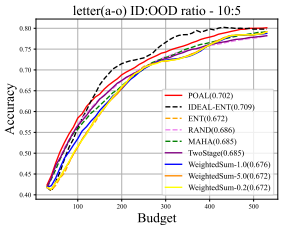
(d) letter(a-l)



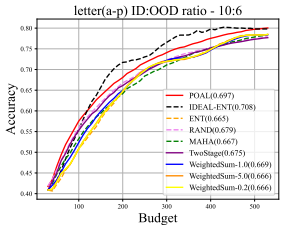
(e) letter(a-n)



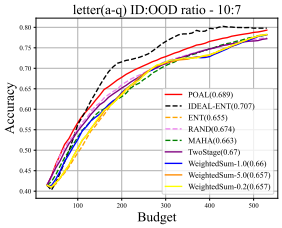
(f) letter(a-m)



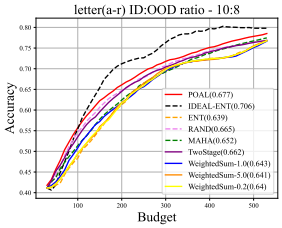
(g) letter(a-o)



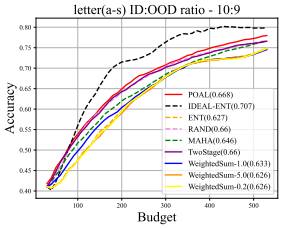
(h) letter(a-p)



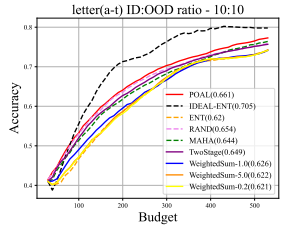
(i) letter(a-q)



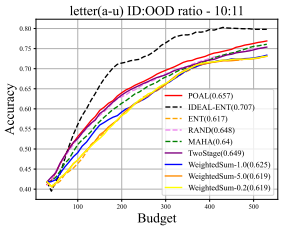
(j) letter(a-r)



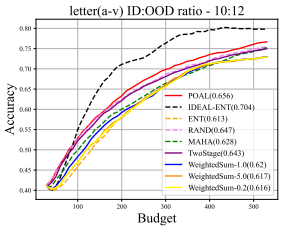
(k) letter(a-s)



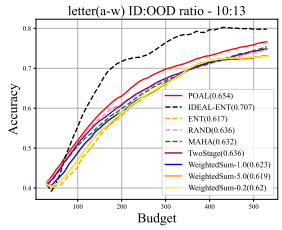
(l) letter(a-t)



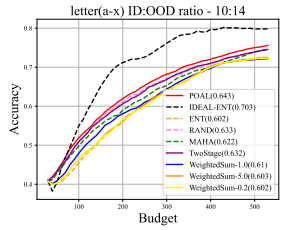
(m) letter(a-u)



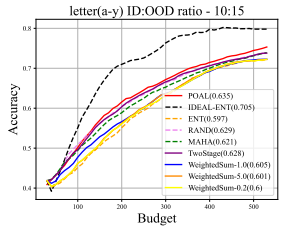
(n) letter(a-v)



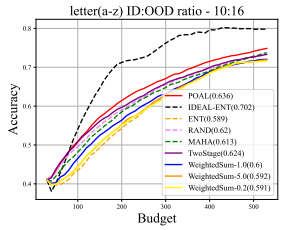
(o) letter(a-w)



(p) letter(a-x)

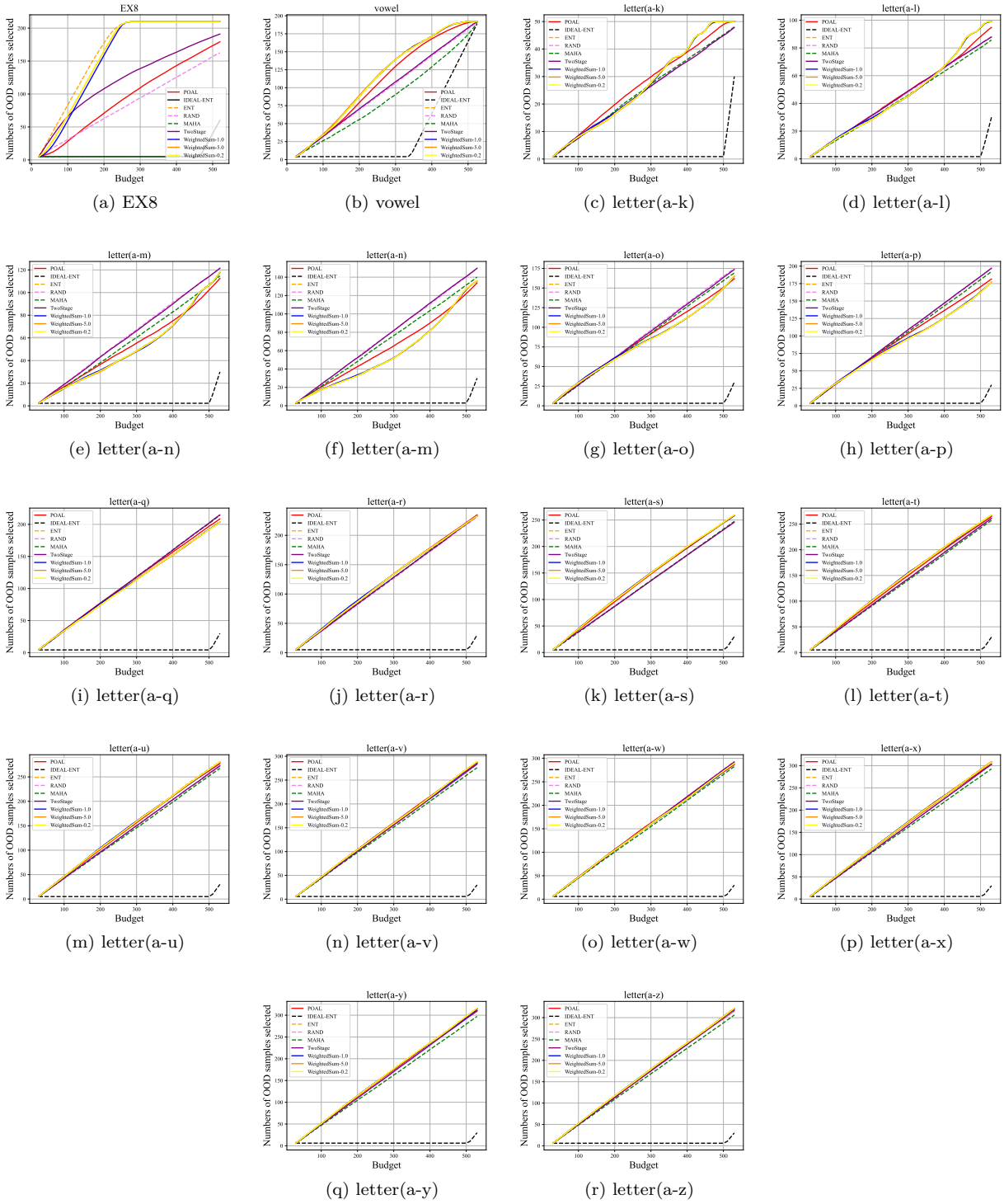


(q) letter(a-y)



(r) letter(a-z)

Figure 8: Accuracy vs. budget curves for classical ML tasks. The AUBC performances are shown in parentheses in the legend.



(a) EX8 (b) vowel (c) letter(a-k) (d) letter(a-l)

(e) letter(a-n) (f) letter(a-m) (g) letter(a-o) (h) letter(a-p)

(i) letter(a-q) (j) letter(a-r) (k) letter(a-s) (l) letter(a-t)

(m) letter(a-u) (n) letter(a-v) (o) letter(a-w) (p) letter(a-x)

(q) letter(a-y) (r) letter(a-z)

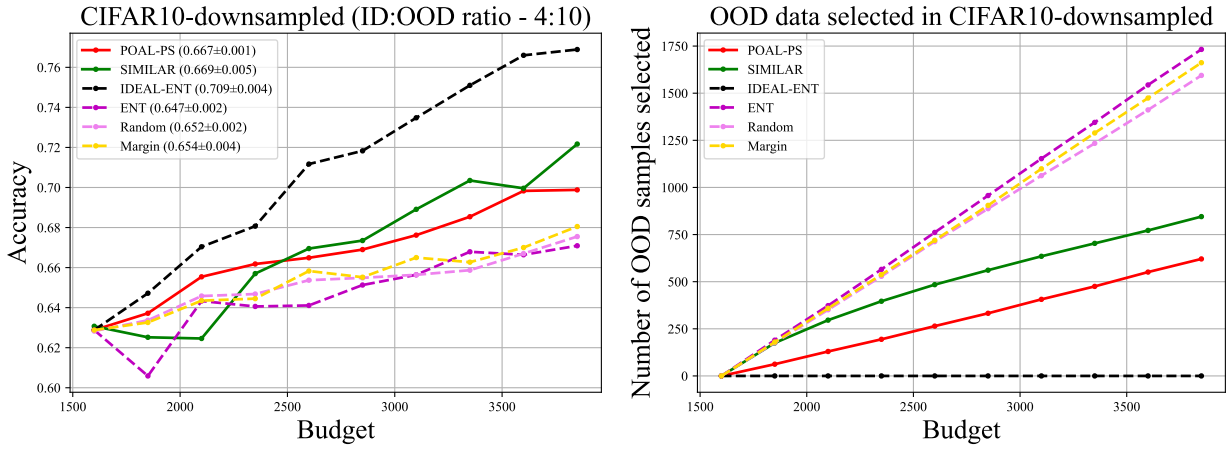
Figure 9: Numbers of OOD samples selected vs. budget curves for classical ML tasks during AL processes.

C.2 POAL vs. SIMILAR (FLCMI)

In the main paper, we have compared the SIMILAR with FLVMI as submodular mutual information. Referring to its comparable performance on *CIFAR10* dataset (see Figure 5a and Figure 5b in (Kothawade

et al., 2021) as reference). We choose FLVMI as baselines in our main paper since it is both time and memory efficient. However, FLVMI performs worse than FLCMI, since FLCMI is specially designed for AL under OOD data scenarios. However, FLCMI is both time and memory-consuming. In (Kothawade et al., 2021), they adopted down-sampled experiments. Although the partition trick could be applied to solve this time/memory-consuming problem, the partition will influence the final performance due to the randomness. To provide a fair comparison. We conduct additional experiments on the down-sampled CIFAR10 dataset on SIMILAR (FLCMI), following the experimental settings in (Kothawade et al., 2021).

The comparison experiments are shown in Fig. 10. Besides our POAL and SIMILAR (FLCMI), we also provide more baselines as reference, i.e., IDEAL-ENT, ENT, Margin (Wang & Shang, 2014) and RAND. As shown in Fig. 10, both SIMILAR (FLCMI) and POAL have better performance than normal AL sampling strategies. From the aspect AUBC evaluation metric, our model has comparable performance with SIMILAR, 0.667 vs 0.669. Nevertheless, we have a lower standard deviation value than SIMILAR, so our POAL is more stable. From the aspect of Accuracy vs. Budget curves, in early stages (e.g., Budget $< 2{,}500$), our POAL is better than SIMILAR (FLCMI) and in latter stages SIMILAR (FLCMI) exceeds POAL. The reason is that in SIMILAR (FLCMI), they calculate the similarities between the ID labeled set and the unlabeled pool and the dissimilarities between the OOD labeled set and the unlabeled pool. In the early stages, the OOD data is insufficient. Thus SIMILAR(FLCMI) will select more OOD samples, as shown in Fig. 10-b. From this experiment, we find that SIMILAR (FLCMI) only performs well when we have enough information on both ID and OOD data samples, which results in more OOD data sample selection. Our POAL only considers the distance between unlabeled samples and labeled ID samples, so we are more efficient in preventing OOD sample selection. Additionally, our POAL is more widely adopted since we could be adopted on large-scale datasets. However, SIMILAR is limited by the computation condition. Our method is also more time efficient than SIMILAR (FLCMI), as shown in Table 4. SIMILAR (FLCMI) runs five times longer than our POAL.



(a) Accuracy vs. Budget

(b) OOD num vs. Budget

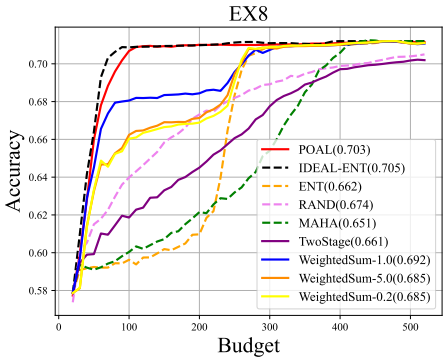
Figure 10: The comparative experiments between our model and **SIMILAR** on down-sampled *CIFAR10* dataset.

Table 4: The mean and standard deviation of running time (in seconds) of the comparative experiments with 3 repeated trials between our POAL and SIMILAR.

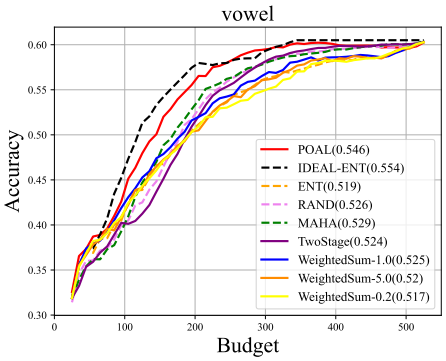
Method	POAL-PS	SIMILAR	IDEAL-ENT	ENT	Random	Margin
Time	6419.0 (109.9)	32837.7 (897.7)	2225.0 (12.6)	1690.0 (15.3)	1507.7 (5.8)	1573.7 (9.8)

C.2.1 Enlarged Figures

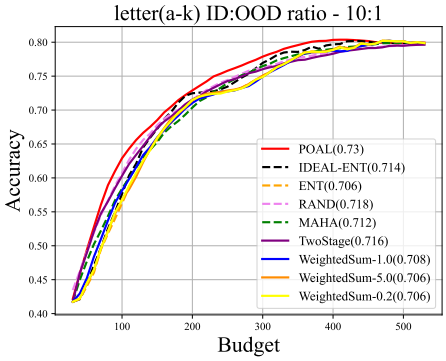
To increase the readability of our experimental results, we enlarged the Figures 3, 4, 5 and 6, as shown in Figures 11, 12, 13 and 14.



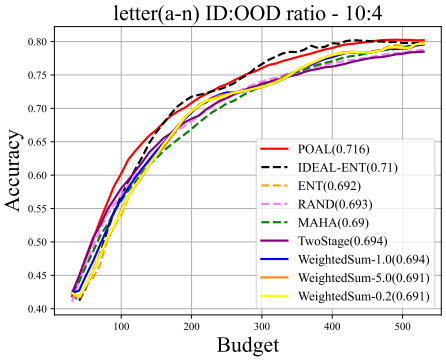
(a) EX8



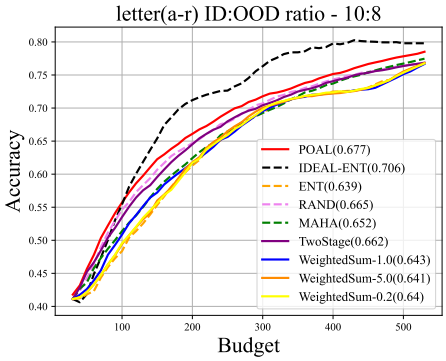
(b) vowel



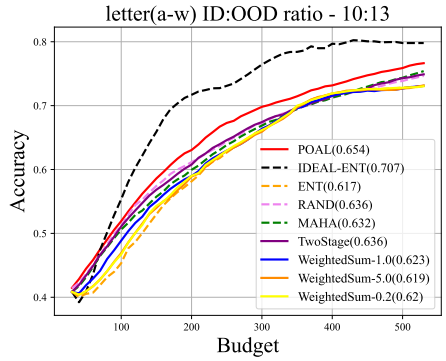
(c) letter(a-k)



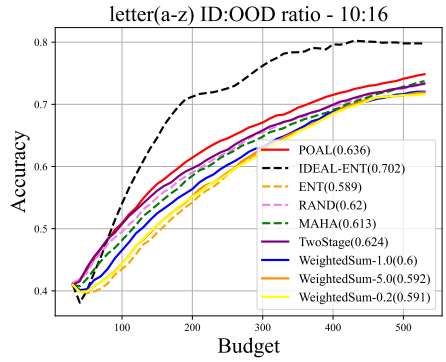
(d) letter(a-n)



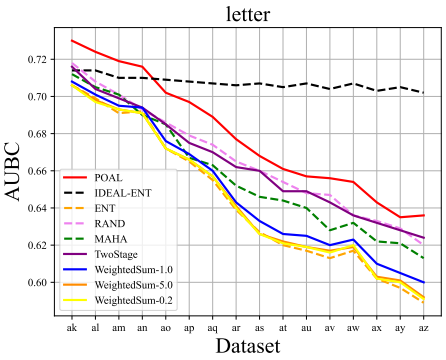
(e) letter(a-r)



(f) letter(a-w)

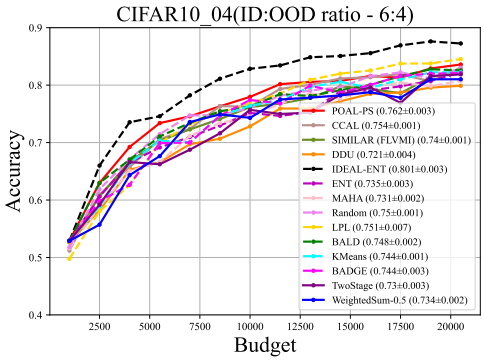


(g) letter(a-z)

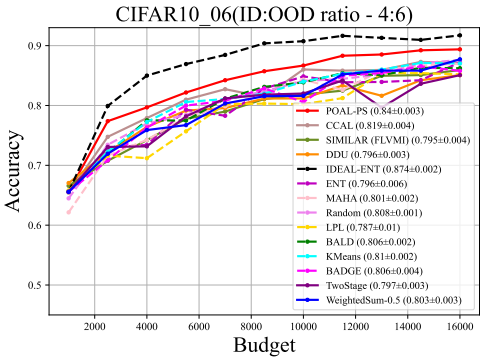


(h) letter(AUBC)

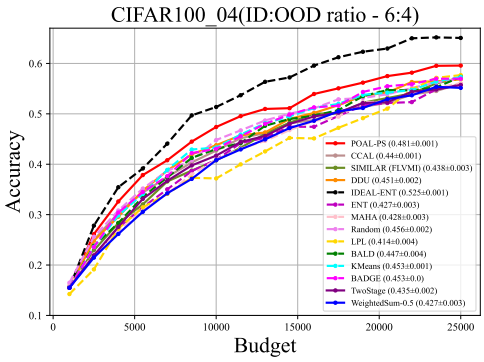
Figure 11: This is a enlarged version of Figure 3 in our main paper.



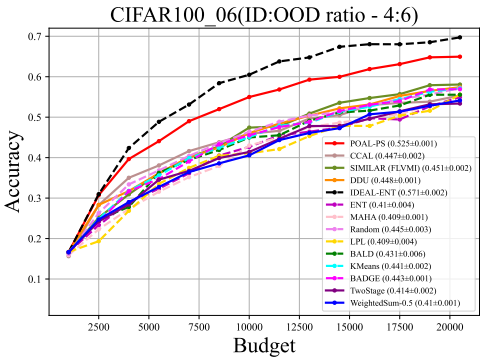
(a) CIFAR10-04



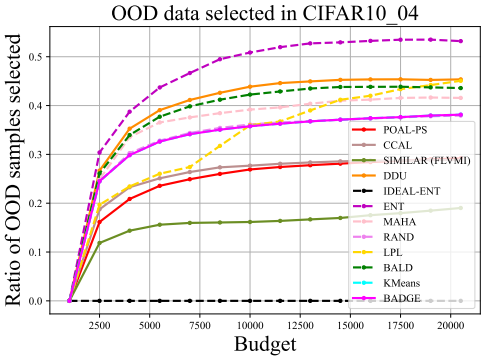
(b) CIFAR10-06



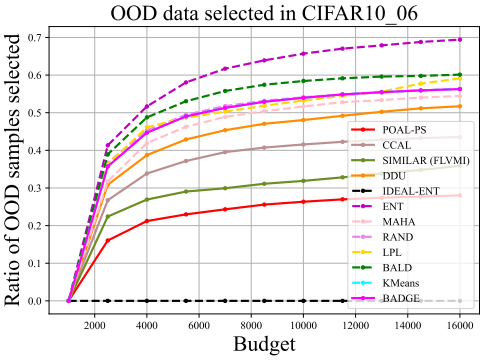
(c) CIFAR100-04



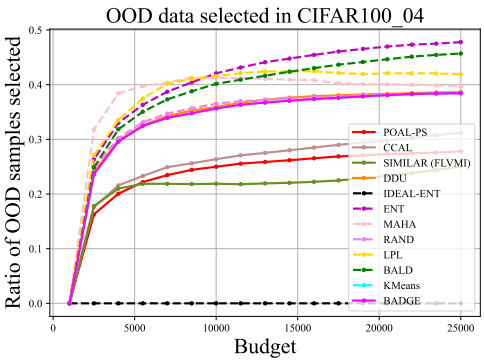
(d) CIFAR100-06



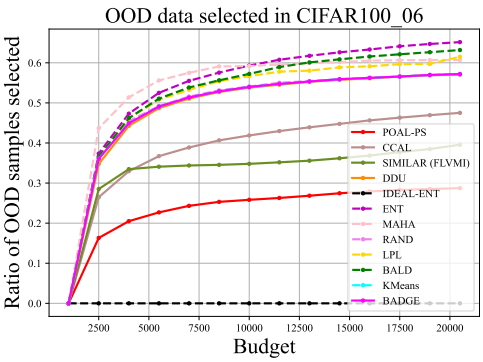
(e) CIFAR10-04 OOD



(f) CIFAR10-06 OOD

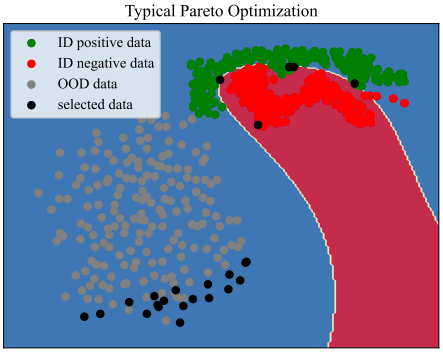


(g) CIFAR100-04 OOD

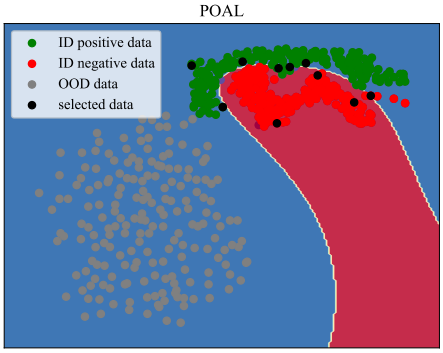


(h) CIFAR100-06 OOD

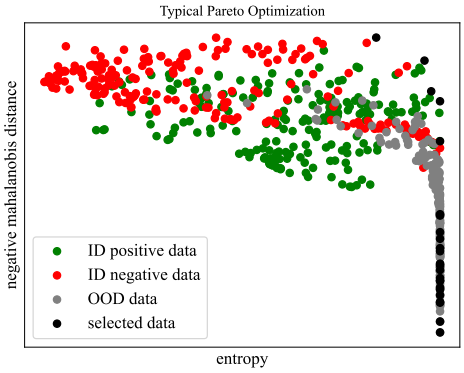
Figure 12: This is a enlarged version of Figure 4 in our main paper.



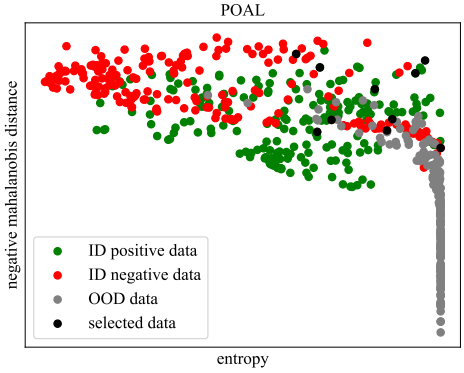
(a) Typical Pareto Optimization



(b) POAL

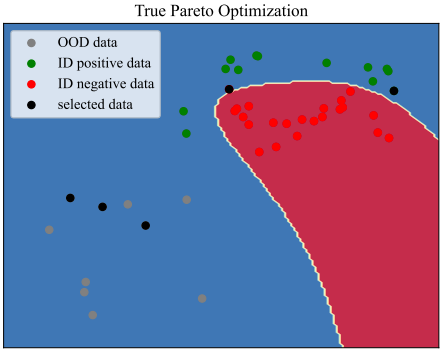


(c) Typical Pareto Optimization

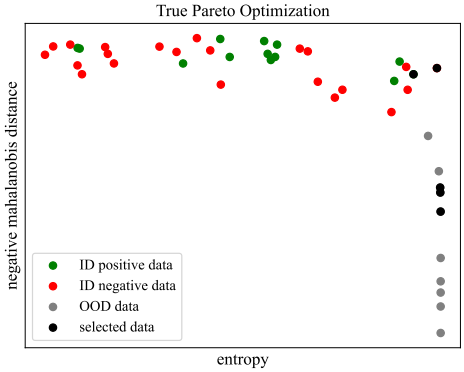


(d) POAL

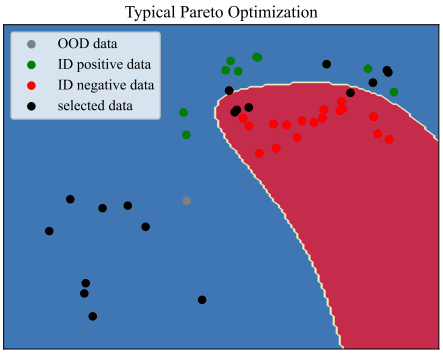
Figure 13: This is a enlarged version of Figure 5 in our main paper.



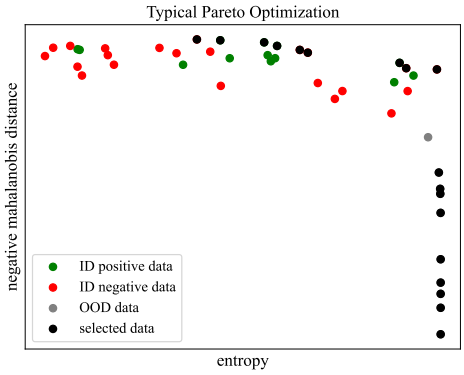
(a) True Pareto Optimization



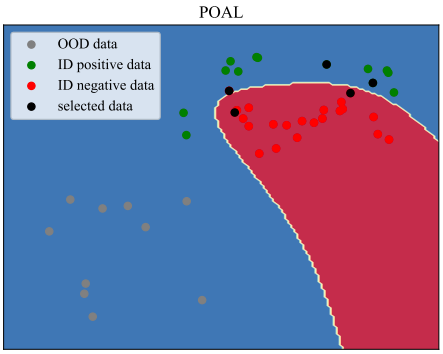
(b) True Pareto Optimization



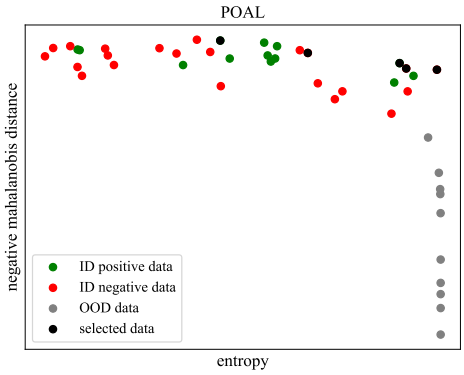
(c) Typical Pareto Optimization



(d) Typical Pareto Optimization



(e) POAL

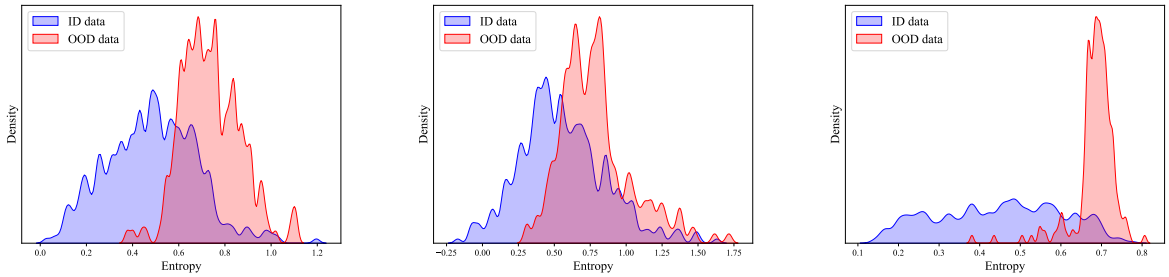


(f) POAL

Figure 14: This is a enlarged version of Figure 6 in our main paper.

C.2.2 Supplementary Experiments vs. Stochastic Acquisition

To explore whether the stochastic acquisition strategy (Kirsch et al., 2021) can be used for AL under OOD detection scenarios, we consider a simple experiment using the same experimental settings as Figure 1. We add Gumbel-distributed random perturbations based on $\mathcal{G}(0; \beta^{-1})$ to the AL score function ENT. The visualization of densities with different β settings are shown in Figure 15. Adding random perturbations could indeed increase the number of ID samples selected, compared with selection with only ENT. Larger β settings could result in better performance, but it still could not distinguish ID and OOD samples, as there is still significant overlap between the ID and OOD score distributions.



(a) $\beta^{-1} = 0.1$ (b) $\beta^{-1} = 0.05$ (c) $\beta^{-1} = 0.02$

Figure 15: Densities when adding Entropy score with Gumbel-distributed random perturbations with different hyper-parameter settings. The experimental settings are the same as Figure 1.