One-Round Active Learning

Tianhao Wang
Harvard University
Cambridge, MA
tianhaowang@fas.harvard.edu

Si Chen
Virginia Tech
Blacksburg, VA
chensi@vt.edu

Ruoxi Jia
Virginia Tech
Blacksburg, VA
ruoxijia@vt.edu

Abstract

Active learning has been a main solution for reducing data labeling costs. However, existing active learning strategies assume that a data owner can interact with annotators in an online, timely manner, which is usually impractical. Even with such interactive annotators, for existing active learning strategies to be effective, they often require many rounds of interactions between the data owner and annotators, which is often time-consuming. In this work, we initiate the study of one-round active learning, which aims to select a subset of unlabeled data points that achieve the highest utility after being labeled with only the information from initially labeled data points. We propose DULO, a general framework for one-round active learning based on the notion of data utility functions, which map a set of data points to some performance measure of the model trained on the set. We formulate the one-round active learning problem as data utility function maximization. We further propose strategies to make the estimation and optimization of data utility functions scalable to large models and large unlabeled data sets. Our results demonstrate that while existing active learning approaches could succeed with multiple rounds, DULO consistently performs better in the one-round setting.

1. Introduction

The success of deep learning largely depends on access to large labeled datasets; yet, annotating a large dataset is often costly and time-consuming. Active learning has been the main solution for reducing labeling costs. The goal of active learning is to select the most informative points from unlabeled data set to be labeled, so that the accuracy of the classifier increases quickly as the set of labeled examples grows. Active learning [8, 9, 10, 26, 5, 23, 28] typically works in multiple rounds, where each round builds a classifier based on the current training set, and then selects the next example to be labeled. The labeled example will then be used to retrain the model and this procedure is repeated until we reach a good model or exceed the labeling budget.

The multi-round nature of existing active learning protocol poses great challenges to its wide adaption. Firstly, it is usually difficult to find annotators who can interact online with the data owner and provide timely feedbacks. For instance, current broadly used data labeling approaches, such as crowdsourcing (e.g., Amazon Mechanical Turk) and outsourcing to companies, do not support a timely interaction between the data owner and annotators. Second, even if one can find such interactive annotators, it is often inefficient to acquire labels one-by-one, especially when each label takes substantial time or when the training process is time-consuming. Various batch-mode active learning algorithms [12, 24, 2, 14, 29, 13] have been proposed to improve labeling efficiency by selecting multiple examples to be labeled in each round. However, for these algorithms to be effective, they still require many rounds of interaction, where the examples selected in one round depend on the examples selected in all previous rounds. Hence, existing batch-mode algorithms cannot make full advantage of parallel labeling.

In light of the aforementioned challenges of multi-round active learning, we argue that a *one-round active learning* is a more realistic and efficient setting, where only one round of data selection and label query is allowed. Specifically, one-round active learning starts with very few labeled data points (e.g., a small amount of randomly selected data that are labeled internally or externally). By exploiting these initially labeled data points, a one-round active learning strategy selects a large subset of unlabeled data points that could potentially achieve high data utility after being labeled, and then queries their labels *all at once*. Importantly, in the one-round setting, the selected data points can be labeled completely in parallel.

In this paper, we propose a general one-round active learning framework via **D**ata **U**tility function **L**earning and **O**ptimization (DULO). Our framework is grounded on the notion of data utility functions, which map any given set of points to some performance measure of the model trained on the set. We propose a natural formulation of the one-round active learning problem, which seeks for the set of

unlabeled points maximizing the data utility function. We adopt a sampling-based approach to learn the data utility function. Our approach is enabled by our empirical observations that most widely-used ML algorithms' data utility functions are close to submodular functions, which can be learned efficiently using samples [4]. We leverage a stochastic greedy algorithm to optimize the data utility function. We further present strategies to make learning and optimization of the data utility function efficient for large models and large unlabeled dataset. Specifically, to improve efficiency for learning the data utility function for large models, we propose to learn it based on a simple proxy model instead of the original model; to improve efficiency for optimizing the data utility function for large unlabeled data, we propose to perform greedy optimization over a block of unlabeled dataset instead of the entire set. Our evaluation shows that our algorithm compares favorably to state-of-art batch active learning strategies in the one-round setting on different datasets and models.

2. Related Work

Earlier studies of active learning focused on the setting where only one example is selected in each round [8, 9, 10, 26, 5, 23, 28]. These methods could be time-consuming for complex models (e.g., deep neural networks), as the model needs to be re-trained after each round. Batch active learning [12] was later proposed to improve the efficiency of label acquisition by querying multiple examples in each round. However, their selection strategy fails to capture the diversity of examples in a batch, thereby often ending up picking redundant examples. Recent works on batch active learning have attempted to enhance the diversity of a batch. For instance, core-set [24] uses k-center clustering to select informative data points while preserving the geometry of data points. However, it only works well for simple data or when the model has good architectural priors. More recently, BADGE [2] proposes to sample a batch of data points whose gradients have high magnitude and diverse directions. This method relies on partially trained models to acquire hypothesized labels and further rank unlabeled data for selection. In the early stage of active learning, the labeled set is still small and the predictions of the model could be unreliable; therefore, the performance of this method may suffer.

Previous works on batch active learning has utilized submodular selection criterion to enable efficient selection over large unlabeled dataset. For example, Batchbald [14] prioritizes data points by estimating both the mutual information between model parameters and a data set and the correlation of data points within a set. The proposed optimization objective is submodular, thereby yielding an efficient greedy algorithm. However, similar to [2], hypothesized labels are used and could be a limitation when the model is not properly trained. On the other hand, [29] and [13] utilized predefined submodular utility functions to both capture the diversity and improve optimization efficiency. [29] measures data informativeness via a classifier's uncertainty [18]. In each round of the algorithm in [29], data points with low uncertainty under the current model are sifted out. To select a subset of diverse samples, the algorithm applies a predefined submodular utility function to the remaining data pool. The algorithm is shown to perform well on multiple classifiers, including deep nets. GLISTER-ACTIVE [13] jointly performs subset selection and training by maximizing the log-likelihood on the labeled subset and it is shown to achieve robust selection results on noisy and imbalanced data. However, [13] also uses hypothesized labels, thus sharing the same limitation as [29] and [13]. In addition, [29] does not evaluate the robustness of their approach on noisy or imbalanced data, which are common in real-world applications.

While the recent works on batch active learning show promising results for choosing diverse data with multiple-round selection, our experiments show that with one-round selection, even the state-of-the-art techniques cannot attain satisfactory performance.

3. Data Utility Functions

3.1. Formalization

A *data utility function* is a mapping from a set of data points to a real number indicating the utility of the set. The metrics for utility could be context-dependent. For instance, in active learning, the goal is often to select a set of unlabeled points that can lead to a classifier with highest test accuracy after labeling. Hence, a natural choice for data utility is the test accuracy of the classifier trained on the dataset.

More formally, we define data utility function in terms of the notion of learning algorithm and metric function. A learning algorithm $\mathcal{A}$ is a function that takes a training set $S = \{(x_i, y_i)\}_{i=1}^n$ and outputs a classifier f. A metric function u takes a classifier as input and outputs model utility. If we define model utility as test accuracy, the metric function is defined as $u(f, \mathcal{V}) = \frac{1}{|\mathcal{V}|} \sum_{(x,y)\in\mathcal{V}} \mathbb{1}[f(x) = y]$ for a test set $\mathcal{V}$ drawn from the same distribution as the training set. However, test set $\mathcal{V}$ is usually not available during the training time. In practice, $u(f, \mathcal{V})$ is usually approximated by *validation accuracy* $u(f, V)$ where V is a validation set separated from the training set. With a learning algorithm $\mathcal{A}$ and a corresponding metric function u, we define the data utility function as $U_{\mathcal{A},u}(S) = u(\mathcal{A}(S), \mathcal{V})$.

3.2. Approximate Submodularity

With the notion of data utility functions, one can conveniently formulate an active learning problem as an optimization problem that seeks the set of unlabeled data points

maximizing the data utility function after being labeled. However, this formulation of active learning requires us to be able to learn and optimize the data utility function efficiently without label information. The data utility function is a set function; a general set function is not efficiently learnable [4] and at the same time, optimizing a general set function is NP-hard [7].

We perform experiments to examine data utility functions for widely used classification models such as logistic regression and our goal is to see if data utility functions exhibit special characteristics that can be leveraged to accelerate the learning and optimization of these functions. Figure 1 shows an example of the data utility function when training set size varies. To produce this figure, we start with an empty training set, randomly sample the training set, and add a data point to the training set each time, and record the model performance. Figure 1 (a) demonstrates the trend of model accuracy with increasing training size for three different random seeds and Figure 1 (b) shows the result averaged over multiple random seeds. A clear "diminishing returns" phenomenon can be observed from these figures, in which an extra training sample contributes less to model accuracy as the training size increases. Moreover, this phenomenon is observed on *almost* every common ML algorithm we test, including complicated models like deep neural networks. Due to space constraint, we defer more examples of such data utility function plots on different datasets and models into the Appendix.

In mathematics, the functions with such a diminishing return property are referred to as *submodular functions*. They have the property that the difference in the incremental value of the function that a single element makes when added into an input set decreases as the size of the input set increases. Formally, a set function $f : 2^V \rightarrow \mathbb{R}$ returning a real value for any subset $S \subseteq V$ is submodular if $f(j \cup S) - f(S) \geq f(j \cup T) - f(T), \forall S \subseteq T, j \in V \setminus T$.

Our experiments show that data utility functions are *approximately* submodular. This empirical finding is fundamental to our framework, because it is known that submodular functions can be learned and optimized efficiently. Specifically, it is shown in [4] that monotonic submodular functions are PMAC-learnable with polynomially many samples if each elements in the subset are selected independently. Moreover, submodular functions can be optimized efficiently by simple greedy algorithms.

In fact, submodularity has been frequently explored by active learning to improve the efficiency of data selection [29, 13, 14]. However, they have mostly choose a surrogate for the true data utility function, such as submodular information-theoretic measures or data utility functions for a simple model which are proven to be submodular, and select the data based on the predefined surrogate and hypothesized labels. Our work differs from these existing endeavors in that we learn data utility function from labeled samples, which is expected to more closely reflect the true utility function and in turn, leads to better data selection results.

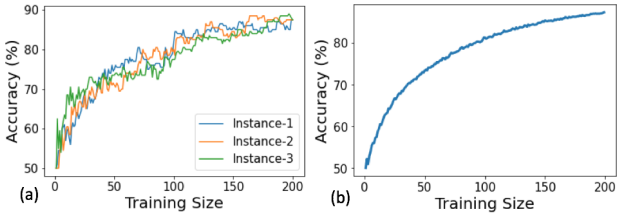

Figure 1: An Illustration of Data Utility Functions for a synthetic dataset (description in Appendix): (a) We randomly sample and add a data point to the training set each time, and record the logistic regression accuracy with the increase of training data size. This plot closely aligns with the definition of submodularity. (b) We sample 50,000 subsets from a dataset of 200 data points and record the test accuracy of a logistic regression trained on each subset. We then plot the mean accuracy for each subset size.

As a final note, although different models' data utility functions consistently exhibit submodular-like property, the rigorous proof is surprisingly hard. Submodularity has thus far been proved for only two simple classes of classifiers–Naive Bayes and Nearest Neighbors. We leave the rigorous proof of the submodular-like property of more widely used models such as logistic regression and deep nets as important future works.

3.3. Learning with DeepSets

Suppose that we have N labeled samples $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^N$. To learn the data utility function, we first split $\mathcal{L}$ into training set $\mathcal{L}_{tr}$ and validation set $\mathcal{L}_{val}$, where $\mathcal{L}_{tr}$ is used for model training and $\mathcal{L}_{val}$ is used for evaluating the data utility (recall that data utility function is approximated by validation accuracy of the model trained on the data). Specifically, we train a classifier f on a subset $S \subseteq \mathcal{L}_{tr}$ and calculate the validation accuracy of f on $\mathcal{L}_{val}$, which gives us $u(f, \mathcal{L}_{val})$. If we train the classifier f for multiple times with different subsets S, the set $\{(g(S), u(f, \mathcal{L}_{val}))\}$ could serve as a training set for learning $U_{\mathcal{A},u}$, where g is a feature extraction function that extracts useful information from S. We adopt a canonical model architecture for set function learning–DeepSets [30]–as our parametric model for approximating data utility function $U_{\mathcal{A},u}$. DeepSets is a kind of neural networks that are permutation-invariant, making them suitable for modeling set-valued data. We then train a DeepSets model f_{DS} on $\{(g(S), u(f, \mathcal{L}_{val}))\}$ to approximate $U_{\mathcal{A},u}$. Algorithm 1 summarizes our algorithm for data utility function learning. For verbal consistency, in the following text, we refer to $\mathcal{L}_{tr}$ as *labeled training dataset*,

Algorithm 1: DeepSets Data Utility Learning.

input : $\mathcal{L}$ - labeled samples drawn i.i.d. from the same distribution, $\mathcal{A}$ - training algorithm for classifier f, u - metric function, $\mathcal{A}_{DS}$ - training algorithm for DeepSets model f_{DS}, r - train-validation split ratio, T - number of classifiers to be trained, g - feature extraction function.

output: f_{DS} - DeepSets utility function which approximates $U_{\mathcal{A},u}$.

Split $\mathcal{L}$ into training set $\mathcal{L}_{tr}$ and validation set $\mathcal{L}_{val}$ according to ratio r.
Initialize training set for DeepSets utility model $S_{DS} = \emptyset$.
for $t = 1, \ldots, T$ **do**
    Randomly choose a subset $S \subseteq \mathcal{L}_{tr}$.
    Train classifier $f \leftarrow \mathcal{A}(S)$.
    $S_{DS} = S_{DS} \cup \{(g(S), u(f, \mathcal{L}_{val}))\}$.
end
Train $f_{DS} \leftarrow \mathcal{A}_{DS}(S_{DS})$.
return f_{DS}

and $\mathcal{L}_{val}$ as *labeled validation dataset*.

4. One-Round Active Learning via DULO

4.1. Core Algorithm

We now discuss how to apply data utility function learning and optimization to solve the one-round active learning problem in a way scalable to large models and large unlabeled data pool. The full algorithm, which we call DULO, is summarized in Algorithm 2. In a high level, our algorithm first learns the data utility function using the initial labeled dataset and then optimizes the learned data utility function to select the most useful unlabeled data points. In order to use the data utility function to select unlabeled data, the input to the learned data utility function should be an unlabeled set. Thus, when learning the data utility function on the labeled set, although the label information is available, we set feature extraction function g to be independent of label and only use the data feature part to predict the data utility. The learned data utility function f_{DS} predicts the test accuracy of the model trained on a given set of unlabeled data *once they are correctly labeled*. The unlabeled data selection problem can be formalized as follows:

$$\underset{|S|=M, S\subseteq\mathcal{U}}{\arg\max} \; f_{DS}(g(S)) \tag{1}$$

where M is the cardinality constraint for the selected subset. Since most of the data utility functions are empirically shown to be approximately submodular, we could solve (1) efficiently and approximately using a greedy algorithm. We choose to apply stochastic greedy optimization [19] to solve (1) because in our experiments, we find it works best in terms of efficiency and performance.

4.2. Improving Scalability

Directly applying the algorithm above could be slow for large models and large unlabeled data. Next, we show how to improve the scalability of our framework.

Scale to Large Models. Constructing a size-T training set for learning the data utility function requires retraining a model for T times. In experiments, we found that for the data utility function to be learned accurately, we often need to train a classifier thousands of times. The computational costs incurred by this process are acceptable for logistic regression or small CNN models, because the initial labeled dataset is usually very small and it takes minimal time to train one model on it. In our experiment, training 1000 logistic regression models on 300 MNIST images in PyTorch [20] only requires around 30 minutes with NVIDIA Tesla K80 GPU, and training 1000 small CNN models on 500 CIFAR-10 images just takes about 4 hours. Besides, since these model trainings are completely independent to each other, we could accelerate the for loop in Algorithm 1 through parallelization without any additional communication cost.

However, for giant models such as a very deep ResNet [11], training on a small dataset multiple times could still be extremely time- or memory-consuming. We resolve this problem by leveraging another empirical observation: the data utilities on different classifiers are correlated with one another. Figure 2 shows an example of such correlation. As we can see, the logistic regression model's accuracies are positively correlated with the LeNet's accuracies, although the LeNet performs much better than the logistic regression. Based on this observation, we propose to use a *proxy model* when training target models thousands of times on the initial labeled dataset exceeds some time or memory budget. The proxy model should be small enough to be trained efficiently, while still achieving observable performance improvement after training on the labeled dataset. When there are too many classes and classification accuracy improvements are hardly noticeable, we can use less stringent utility metrics functions such as Top-5 accuracy.

Scale to Large Datasets. Another design choice of our one-round active learning framework is to deal with the situation when the size of unlabeled dataset is much larger than the labeled set, i.e. $|\mathcal{U}| \gg |\mathcal{L}_{tr}|$. In this case, the generalizability of DeepSets models to giant input sets is questionable, because the largest possible data size that the DeepSets model has seen during training is $|\mathcal{L}_{tr}|$. Besides, the evaluation time for DeepSets models will significantly

increase for larger input set sizes, which makes the linear-time stochastic greedy algorithm become inefficient in practice. We tackles these challenges by running greedy optimization with a subset $\mathcal{B} \subseteq \mathcal{U}$ (block) of an appropriate size B. B should be chosen such that the DeepSets data utility model has good generalizability and efficient evaluation. We can repeat the above process for $\frac{|\mathcal{U}|}{B}$ times and each time, we select $\frac{MB}{|\mathcal{U}|}$ data points from each of $\mathcal{B}$. We call parameter B as *optimization block size* in Algorithm 2. Although breaking down the selected data into different blocks, we want to highlight that this design is completely different from batch active learning, since we do not require the data points selected in the previous blocks to be labeled before we select data points from the current block. Therefore, the while loop in Algorithm 2 is fully parallelizable. One may concern that choose data points independently from each block would fail to capture the interactions between the data selected from different blocks. However, as long as the optimization block size is reasonably large, the missed high-level interactions are negligible. We perform an ablation study for optimization block size B in Section 5.2.2.

Comparison with Batch Active Learning Strategies. Our approach is similar to FASS [29] and some other popular batch active learning strategies [13] in the sense that we all try to select subsets of unlabeled datasets that optimize a particular criterion, e.g. a submodular function that approximates data utility function. A critical difference between DULO and other approaches is that DULO does not require hypothesized labels from the partially trained classifier for greedy optimization. This could be a significant advantage especially when the amount of initially labeled data points is very small and the classifier trained with these data has very poor performance.

5. Evaluation

In this section, we evaluate the performance of DULO and compare with the state-of-the-art batch active learning strategies in one-round setting.

5.1. Experimental Settings

5.1.1 Evaluation Protocols

We evaluate the performance of different active learning strategies on different types of models and a varied amount of selected data points. Notably, we assess the performance of DULO for models that are more complicated than proxy model. For example, we use logistic regression as the proxy model for MNIST, while examine the utility of the selected data for LeNet. We evaluate the performance of DULO on different practical scenarios such as imbalanced or noisy

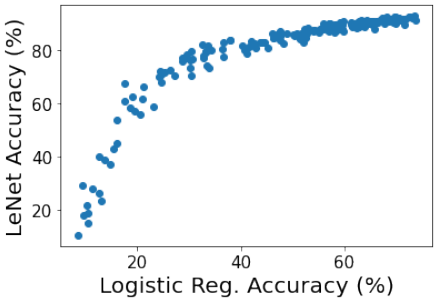

Figure 2: Example of the correlation between data utilities for different classifiers. For each point in the figure, we randomly sample 2000 data points from MNIST, add Gaussian noise to a random portion of the images, and check the accuracies of logistic regression model and LeNet trained on the selected samples.

Algorithm 2: DULO for one-round active learning

input : $(\mathcal{L}, \mathcal{A}, \mathcal{A}_{DS}, u, r, T, g)$ - same inputs as Algorithm 1, feature extraction function g only depends on data information, $\mathcal{U}$ - unlabeled samples, M - number of data to be selected, B - optimization block size

output: $S_{selected}$ - selected set.

$f_{DS} \leftarrow$ DeepSetsDataUtilityLearning$(\mathcal{L}, \mathcal{A}, \mathcal{A}_{DS}, u, r, T, g)$

Initialize $S_{selected} = \emptyset$.

while $|S_{selected}| < M$ **do**

  Randomly choose a subset $\mathcal{B} \subseteq \mathcal{U}$ where $|\mathcal{B}| = B$.

  Find $S \in \arg\max_{|S|=\frac{MB}{|\mathcal{U}|}, S \subseteq \mathcal{B}} f_{DS}(g(S))$ with stochastic greedy optimization.

  $S_{selected} = S_{selected} \cup S$.

  $\mathcal{U} = \mathcal{U} \setminus S$.

end

return $S_{selected}$

unlabeled datasets. We also test the performance on real-world datasets and provide insights into the selected data. Finally, we perform ablation studies for sample complexity and different hyperparameters for DULO. We also studied the potential variant of DULO combining with uncertainty sampling and the results are presented in the Appendix.

5.1.2 Datasets & Implementation Details

We evaluate the robustness of DULO on class-imbalanced and noisy unlabeled datasets from MNIST [16] and CIFAR-10 [15]. We also perform experiments on USPS [1] and PubFig83 [22] to demonstrate effectiveness of DULO on real-world datasets. We summarize the sizes of labeled

Dataset	$\lvert\mathcal{L}_{tr}\rvert$	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{L}_{val}\rvert$	Proxy Model
MNIST	300	2000	300	Logistic Reg.
CIFAR-10	500	20000	500	SmallCNN
USPS	300	2000	300	Logistic Reg.
PUBFIG83	500	10000	500	LeNet

Table 1: Dataset Settings.

training set ($\mathcal{L}_{tr}$), unlabeled set ($\mathcal{U}$), labeled validation set ($\mathcal{L}_{val}$), and proxy model in Table 1. All of the $\mathcal{L}_{tr}$ and $\mathcal{U}$ are sampled from the corresponding datasets' original training data. The sampling distribution is uniform unless otherwise specified in the paper. $\mathcal{L}_{val}$ are uniformly drawn from the corresponding datasets' original test data. For all datasets, we randomly sample 4000 subsets of $\mathcal{L}_{tr}$ and use the corresponding proxy model to generate the training data for utility learning. For each subset sampling, we first uniformly randomly generate a real value $\alpha \in [80, 100]$ for each class, and then draw a distribution sample p from Dirichlet distribution $Dir(\alpha_1, \ldots, \alpha_K)$ where K represents the number of classes. We draw a subset with different class sizes proportional to p. This sampling design is to encourage the diversity of class distributions in the sampled subsets.

For feature extraction function g, we use LeNet model [17] to extract the image features for MNIST, and we use ImageNet-pretrained Inception-V3 [27] to extract image features for CIFAR-10 and PubFig83 as the input for DeepSets utility models. We set $\epsilon = 10^{-5}$ in stochastic greedy optimization for line 14 in Algorithm 2, and we set the optimization block size $B = 2000$ in all cases. We select up to half of the data points from $\mathcal{U}$ in the one-round active learning. We test DULO and baselines on five different labeled/unlabeled dataset pairs for every experiment. For each labeled/unlabeled dataset pair, we repeat DULO and other baseline algorithms ten times and take the average of the model accuracies. We defer the dataset descriptions and the architectures, hyperparameters, and training details of proxy models and DeepSets utility models to the supplementary materials.

5.1.3 Baseline Algorithms

We consider several state-of-the-art batch active learning strategies as our baselines: **(1) FASS** [29] first filters out the data samples with low uncertainty about predictions. It then selects a subset by first assigning each unlabeled data a hypothesized label with the current partially-trained classifier's prediction and optimizing a Nearest Neighbor submodular function on the unlabeled dataset with hypothesized labels.[1] **(2) BADGE** [3] first generates hypothesized labels and selects a subset based on the diverse gradient embedding obtained with the hypothesized samples. **(3) GLISTER** [13] also generates hypothesized labels and is formulated as a discrete bi-level optimization problem on the hypothesized samples. **(4) Random** is a setting where we randomly select a subset from the unlabeled datasets.

FASS, BADGE, GLISTER are initially designed for multi-round active learning, but they can be trivially extended to the one-round setting by limiting the number of data selection rounds to be 1. Since these methods have been shown to outperform techniques like uncertainty sampling [25] and coreset-based approaches [24], we do not compare them in this work. We test these baselines with open-source implementation[2], and we defer the hyperparameter settings for these baselines to the Appendix.

5.2. Experiment Results

5.2.1 Data Selection Performance

We evaluate the performance of DULO for one-round active learning in different dataset settings, including class-imbalanced, noisy, and real-life datasets. For MNIST, we show the accuracies of both logistic regression (MNIST's proxy model) and LeNet model on the selected data. For CIFAR-10, since the accuracy of its proxy model is relatively low (around 30%) in general, we only show the data selection results for a large CNN model (denoted as LargeCNN in figures). For USPS dataset, we test the active learning performance on logistic regression (USPS's proxy model) and SVM. For PubFig83, since it has 83 classes in the range domain, we show the top-5 accuracies of LeNet model (PubFig83's proxy model) and a large CNN model. In Appendix, we describe the architectures and training details for these test models as well as the experiment results on other classifiers.

Imbalanced Dataset. We artificially generate class-imbalance for MNIST unlabeled dataset by sampling 55% of the instances from one class and the rest of the instances uniformly across the remaining 9 classes. For CIFAR-10's unlabeled dataset, we sample 50% of the instances from two classes, 25% instances from another two classes, and 25% instances from the remaining 6 classes. The results for MNIST class-imbalance setting are shown in Figure 3 (a) and (b), which are the accuracies of logistic regression (MNIST's proxy model) and LeNet on the selected datasets, respectively. The results demonstrate that DULO significantly outperforms the other baselines. By examining the data points selected by different strategies, we found that DULO can select more balanced dataset, thereby leading to higher performance. We note that DULO's improvement

[1] We do not compare to the Naive Bayes submodular function since NN submodular function mostly outperforms NB submodular for non NB models, as demonstrated in the original paper.

[2] https://github.com/decile-team/distil

is smaller for LeNet, likely because Convolutional Neural Network is more robust to imbalanced training data than logistic regression. Figure 3 (e) shows the performance of class-imbalance one-round active learning on CIFAR-10. Again, DULO consistently outperforms other baselines.

Noisy Dataset. To create noisy datasets, we inject Gaussian noise into images in random subsets of data. The noise scale refers to the standard deviation of the random Gaussian noise. For MNIST, we add noise to 75% unlabeled data points with noise scale 1.0. For CIFAR-10, we add noise to 25% unlabeled data points with noise scale 1.0. Figure 3 (c) (d) and (f) show the results for noisy data settings. We observe that all three baselines perform very differently on MNIST and CIFAR-10. In contrast, DULO outperforms other baselines by more than 1% for every data selection size, and is very close to, or sometimes even better than the optimal curves, which are calculated by taking an average over accuracies of noiseless datasets. To get more insights into the selection process, we add each of the noise scales of 0.25, 0.6, and 1.0 to 25% unlabeled data (i.e., 75% of the unlabeled data are noisy and there are 3 different noisy levels), and we inject an MNIST image together with its three noisy variants of different noise scales into the MNIST unlabeled dataset and examine their selection order, shown in Figure 4 (a). We can see that DULO tends to select clean, high-quality images early and is able to sift out noisy data.

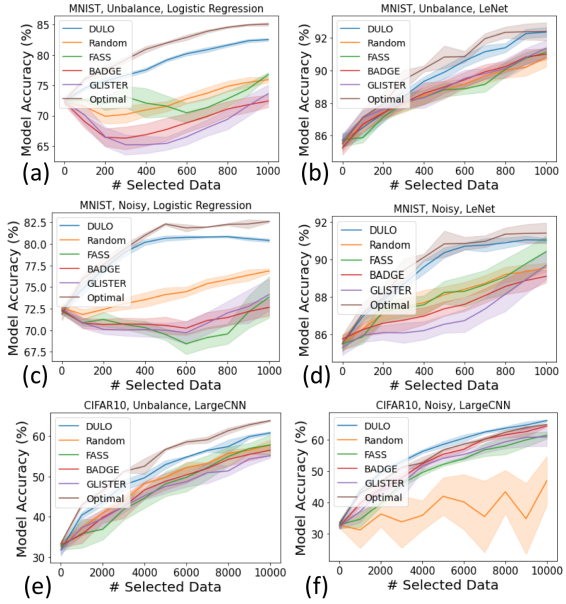

Figure 3: Target models' accuracies vs. the number of selected samples for different settings. Standard errors are shown by shaded regions. The optimal curves are the mean accuracies of balanced/noiseless datasets.

Figure 4: Selection ranks for different images in a greedy optimization block. (a) MNIST (b) USPS (c) PubFig83.

Real-life Datasets. We study the effectiveness of different strategies on real-life datasets, including USPS and PubFig83 datasets, which contain some natural variations in image quality. Figure 5 (a) and (b) show the active learning performance on USPS dataset with logistic regression (proxy model) and support vector machine. For both cases, we can see that DULO consistently outperforms all other baselines. Figure 4 (b) illustrates the images selected in different ranks by DULO. We can see that the points selected earlier have higher image quality; especially, the last five selected '6' digit images tend to be blurry and some could easily be confused with other digits. The last ten selected images contain four '1's, which is because USPS is a class-imbalanced dataset, and there are more '1's than other classes. Figure 5 (c) and (d) show the performance on PubFig83 dataset with LeNet (proxy model) and a large CNN model. Again, for both cases, we can see that DULO consistently outperforms all other baselines. Figure 4 (c) shows the rankings of 3 images within an optimization block. Interestingly, the early face images selected by DULO contain complete face features, while the later ones either contain more irrelevant features or are corrupted.

5.2.2 Ablation Study

Sample Complexity of Data Utility Learning There are two kinds of sample complexity involved in the DeepSets-based utility model learning: (1) *data complexity* which is the size of labeled training dataset $|\mathcal{L}_{tr}|$, and (2) *subset complexity* which is the number of subsets T for the initially labeled dataset we sample to train proxy models. Both quantities need to be appropriately large to train a utility model that can generalize well to unseen subset and capture the interactions between different data points. Figure 6 (a) and (b) show the one-round active learning performance for DeepSets utility models trained with different amount of subset samples. We find that the performance is nearly optimal as long as the number of subset samples is above 500. This is not a large number: training 1000 logistic regression models on MNIST on 300 data points in PyTorch only requires around 30 minutes with NVIDIA Tesla K80 GPU. Since CNN models are more effective on MNIST, even the data utility model trained with only 250 samples is able to achieve comparable performance to the optimal accuracy

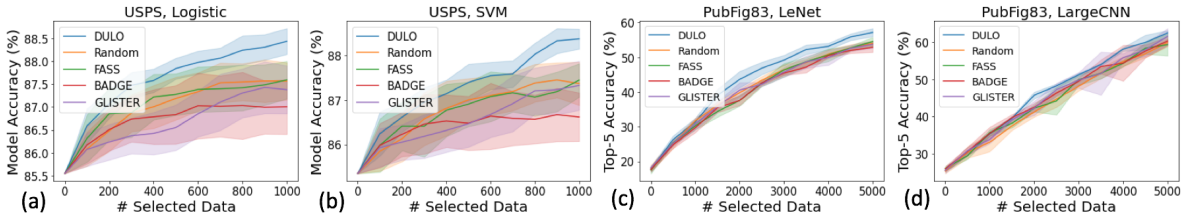

Figure 5: One-round Active Learning on Real-life Datasets.

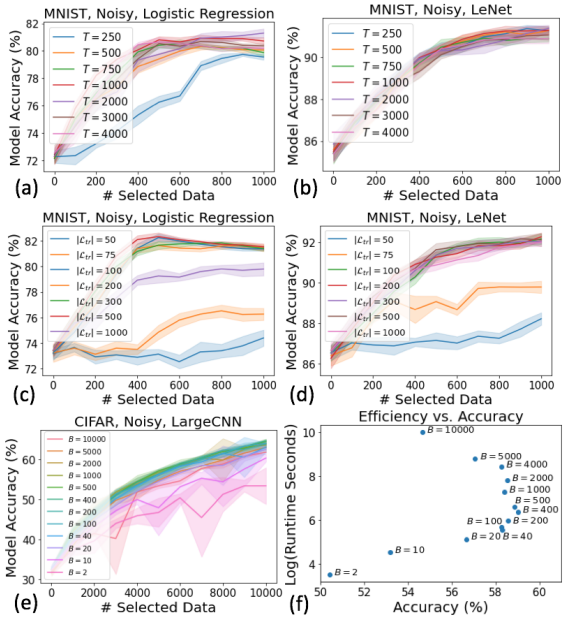

Figure 6: Ablation Study: (a) and (b) studies the subset complexity for utility learning. (c) and (d) studies data complexity for utility learning. (e) compares DULO's performance with different optimization block sizes, and (f) studies the relationship between efficiency and active learning performance for different block sizes when the number of selected data is 6000.

for learning LeNet.

Figure 6 (c) and (d) show the one-round active learning performance for DeepSets utility models trained with different amount of labeled training samples $|\mathcal{L}_{tr}|$. Note that here the largest possible training samples for the proxy model varies, but we still train the classifiers with 300 labeled training samples together with selected data points. We fix the block size and number of subsets sampled for DeepSets training in the experiments. As we can see, near-optimal performance could be achieved even when there are only 100 labeled training data points. When $|\mathcal{L}_{tr}|$ is too large, the trained DeepSets model's performance might degrade due to insufficient training samples compared with input dimensions. When $|\mathcal{L}_{tr}|$ is too small, the generalizability of DeepSets utility model to large sets would be worse and at the same time, the block size has a small upper limit, which can capture less data interaction. A practical guideline for setting the size of initial labeled dataset is in the level of hundreds.

Optimization Block Size Figure 6 (e) shows the one-round active learning performance with different optimization block sizes B. As we can see, both too small and too large block size can degrade the selected data's utility. When B is too small, DeepSets fails to capture interactions between data points selected from different blocks. When B is too large, DeepSets has large generalization error.

Although the stochastic optimization algorithm runs in $O(|\mathcal{U}|)$ regardless of the block size (without parallelization), individual DeepSets evaluation time increases significantly with a larger input set. Figure 6 (f) shows the runtime vs accuracy plot with different block sizes. We can see that when B is too large, it suffers from both poor utility and inefficiency. However, we find a wide range of B that achieves both good efficiency and high data utility (the points located on the right of the figure), which is around $|\mathcal{L}_{tr}|$ ($|\mathcal{L}_{tr}| = 500$ for CIFAR-10 in our setting). This servers as a heuristic choice of B. In this range, the block size is small enough so that DeepSets models generalize well on input sets of block sizes, while also being large enough so that the utility model could capture most of the data interactions.

6. Conclusion

This paper propose a general framework for one-round active learning via data utility function learning and optimization. Our evaluation shows that it outperforms existing baselines across various datasets and models. There are two assumptions underlying our approach: the labels of the initial labeled dataset is correct and the labeled and unlabeled datasets should share the same set of labels. Extending DULO to noisy initial labeled data and varied label settings would be interesting future directions. Moreover, we would like to complement our currently empirical-only results of approximately submodular data utility for various machine learning models with rigorous analysis.

References

[1] E Alpaydin and C Kaynak. Optical recognition of handwritten digits data set. *UCI Machine Learning Repository*, 1998.

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

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

[4] Maria-Florina Balcan and Nicholas JA Harvey. Learning submodular functions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 793–802, 2011.

[5] Colin Campbell, Nello Cristianini, Alex Smola, et al. Query learning with large margin classifiers. In *ICML*, volume 20, page 0, 2000.

[6] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[7] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[8] Shai Fine, Ran Gilad-Bachrach, and Eli Shamir. Query by committee, linear separation and random walks. *Theoretical Computer Science*, 284(1):25–51, 2002.

[9] Yoav Freund, H Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2):133–168, 1997.

[10] Thore Graepel and Ralf Herbrich. The kernel gibbs sampler. In *NIPS*, pages 514–520. Citeseer, 2000.

[11] 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*, pages 770–778, 2016.

[12] Steven CH Hoi, Rong Jin, Jianke Zhu, and Michael R Lyu. Batch mode active learning and its application to medical image classification. In *Proceedings of the 23rd international conference on Machine learning*, pages 417–424, 2006.

[13] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. *arXiv preprint arXiv:2012.10630*, 2020.

[14] Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *arXiv preprint arXiv:1906.08158*, 2019.

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

[16] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[18] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

[19] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[21] 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.

[22] Nicolas Pinto, Zak Stone, Todd Zickler, and David Cox. Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook. In *CVPR 2011 WORKSHOPS*, pages 35–42. IEEE, 2011.

[23] Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *ICML*, volume 2, page 6. Citeseer, 2000.

[24] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

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

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

[27] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[28] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

[29] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pages 1954–1963. PMLR, 2015.

[30] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017.

A. Additional Experiment Settings

A.1. Dataset for Creating Figure 1 in the Main Text

In Figure 1 of the maintext, we showed the data utility function for a synthetic dataset with logistic regression. For the synthetic data generation, we sample 200 data points from 50-dimensional standard Gaussian distribution. All of the 50-dimensional parameters are independently and uniformly drawn from $[-1, 1]$. Each data point is labeled by the sign of its vector's sum.

A.2. Details of Datasets Used in Section 5

MNIST. MNIST consists of 70,000 handwritten digits. The images are 28×28 grayscale pixels.

CIFAR-10. CIFAR-10 consists of 60,000 3-channel images in 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck). Each image is of size 32×32.

USPS. USPS is a real-life dataset of 9,298 handwritten digits. The images are 16×16 grayscale pixels. The dataset is class-imbalanced with more 0 and 1 than the other digits.

PubFig83. PubFig83 is a real-life dataset of 13,837 facial images for 83 individuals. Each image is resized to 32×32.

A.3. Implementation Details

A small CNN model is used as the proxy model for CIFAR-10 dataset, which has two convolutional layers, each is followed by a max pooling layer and a ReLU activation function. LeNet model is the proxy model for PubFig83 dataset, and we also used it to test the active learning performance for MNIST in the experiment. LeNet is adapted from [17], which has two convolutional layers, two max pooling layers and one fully-connected layer. A large CNN model is used to evaluate active learning performance on CIFAR-10 and PubFig83, which has six convolutional layers, and each of them is followed by a batch normalization layer and a ReLU layer. We use Adam optimizer with learning rate 10^{-3}, mini-batch size 32 to train all of the aforementioned models for 30 epochs, except that we train LeNet for 5 epochs when using it for testing active learning performance on MNIST. We also use the support vector machine (SVM) to evaluate active learning performance on the USPS dataset. We implement SVM with scikit-learn [21] and set the L2 regularization parameter to be 0.1.

A DeepSets model is a function $f_{DS}(S) = \rho(\sum_{x \in S} \phi(x))$ where both ρ and ϕ are neural networks. In our experiment, both ϕ and ρ networks have 3 fully-connected layers. For MNIST and USPS, we set the number of neurons to be 128 in each hidden layer. For CIFAR-10 and PubFig83, we set the number of neurons in every hidden layer to be 512. We set the dimension of set features (i.e., the output of ϕ network) for DeepSets models to be 128 in all experiments, except for USPS dataset the set feature number is 64. We use the Adam optimizer with learning rate 10^{-5}, mini-batch size of 32, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ to train all of the DeepSets utility models.

In terms of the baseline batch active learning approaches, we set β in FASS to be the size of unlabeled dataset after parameter tuning. We set the learning rate in GLISTER to be 0.05, following the original paper.

B. Additional Experiment Results

B.1. More Examples of Data Utility Functions

Figure 7 shows more examples of data utility functions for commonly used datasets, including Iris [6], MNIST [16], CIFAR-10 [15], USPS [1], as well as commonly used models such as logistic regression, convolutional neural network and support vector machine. To produce these figures, we start with an empty training set. In each round, we randomly sample a data point from the available training data, add the data point to the training set, and record the test accuracy of the model trained on the current training set. Figure 7a, 7c, 7e, 7g demonstrate the trends of model test accuracy with increasing training size for three different random seeds. We then sample 5000 subsets from each dataset and record the test accuracies of models trained on each subset. Figure 7b, 7d, 7f, 7h show the mean accuracy for different subset sizes. In all these figures, we can observe a clear "diminishing returns" phenomenon, in which an extra training sample contributes less to model accuracy as the training size increases.

B.2. Combining with Uncertainty Sampling

Traditionally, active learning techniques like uncertainty sampling (US) and query by committee (QBC) have shown great promise in several domains of machine learning [25]. However, in the task of multi-round batch active learning, naive uncertainty sampling fails to capture the interactions between selected samples. Simply choosing the most uncertain samples may lead to a selected set with very low diversity. Filtered Active Submodular Selection (FASS) [29] combines the uncertainty sampling method with a submodular data subset selection framework. Specifically, at every round t, FASS first selects a candidate set of β_t most uncertain samples among unlabeled data, and then runs greedy optimization on an appropriate submodular objective (with the hypothesized labels assigned by the current model). It is natural to ask whether we should also combine DULO with uncertainty sampling by only performing greedy optimization on the most uncertain samples. We evaluate the performance of DULO when we first filter out data points that the current classifier has high confidence about, and preserve a candidate set of β most uncertain samples on which we run stochastic greedy optimization. We show the performance of different values of β on class-imbalanced and noisy MNIST dataset in Figure 8, where $\beta = |\mathcal{U}| = 2000$ coincides with the setting of the original DULO algorithm. As we can see, for all settings studied in our experiments, $\beta = 2000$ consistently outperforms other smaller values of β. Hence, using uncertainty sampling to pre-process unlabeled samples does not seem to lead to better performance in our one-round active learning framework. We conjecture that this is because we only have very few labeled samples

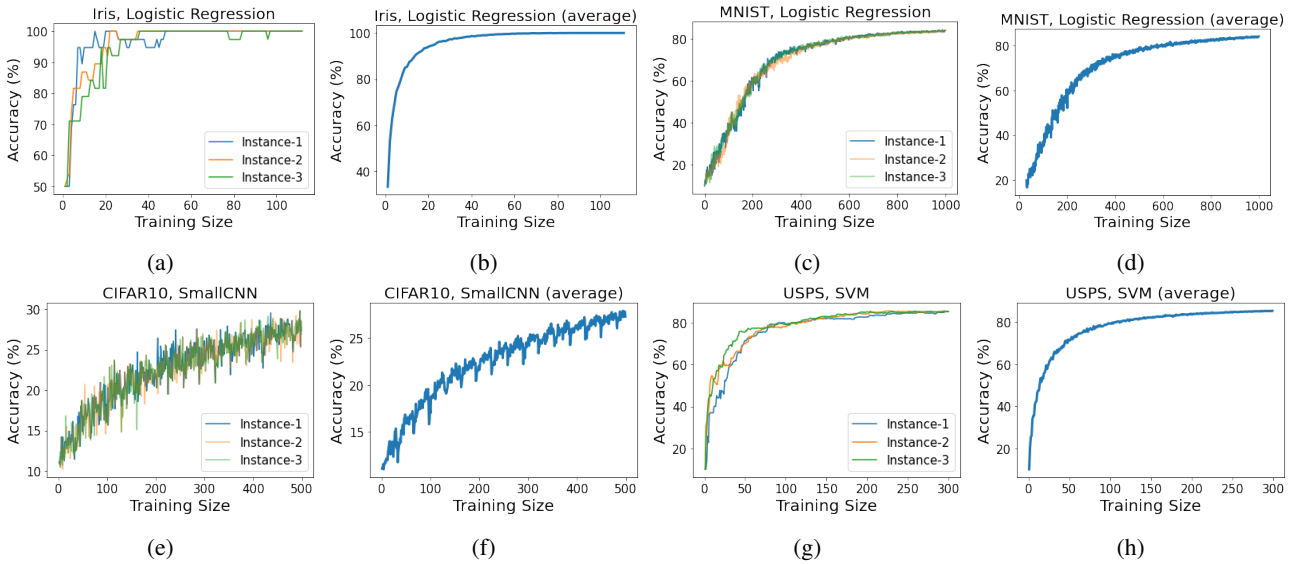

Figure 7: More Examples of Data Utility Functions with different datasets and commonly used models.

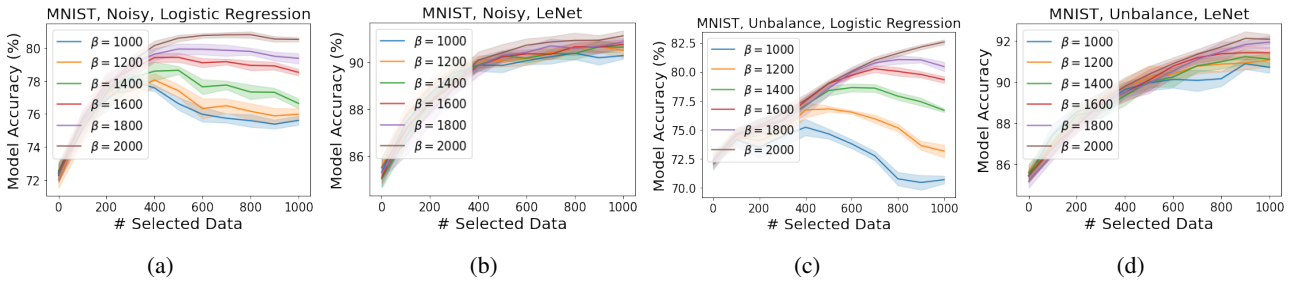

Figure 8: Ablation Study: the performance of DULO when we run greedy optimization on the β most uncertain samples of the current classifiers.

at the beginning. In that case, the performance of the initial classifiers is pretty poor and their uncertainty outputs are not informative.