ZERO-ROUND ACTIVE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Active learning (AL) aims at reducing labeling efforts by identifying the most valuable unlabeled data points from a large pool. Traditional AL frameworks have two limitations: First, they perform data selection in a multi-round manner, which is time-consuming and sometimes impractical. Second, they assume that there are a small amount of labeled data points available in the same domain as the data in the unlabeled pool. We propose $\mathrm{D}^2\mathrm{ULO}$ as a solution that solves both issues. The key observation that enables our approach is that there often exists labeled data from domains (i.e., source domains) related to the unlabeled data's domain (i.e., target domain). Based on this observation, our approach leverages the idea of domain adaptation (DA) to train a data utility model that can effectively predict the utility for any given unlabeled data in the target domain. The trained data utility model can then be used to select high-utility data and at the same time, provide an estimate for the utility of the selected data. Our algorithm can work standalone as a zero-round active learning approach which selects unlabeled data all at once and independently of the human labeler's feedback. It can also work in tandem with existing multi-round active learning approaches by providing a warm-start. Our experiments evaluate both use cases. We first focus on the zero-round AL setting and show that $\mathrm{D}^2\mathrm{ULO}$ outperforms the existing state-of-the-art AL strategies equipped with domain adaptation, a natural baseline for our approach, over various domain shift settings (e.g., real-to-real data and synthetic-to-real data). Particularly, $\mathrm{D}^2\mathrm{ULO}$ is applicable to the scenario where source and target labels have mismatch, which is not supported by the existing works. Moreover, when used as a warm-start, our approach can effectively improve the performance of existing multi-round AL methods.

1 INTRODUCTION

Deep neural networks (DNNs) have been successful on various tasks across different fields with the help of large-scale labeled datasets. However, data labeling processes are often expensive and time-consuming. One popular framework to reduce labeling costs is *active learning* (AL), which strategically selects and labels the data instances from the unlabeled data pool with the goal of achieving comparable performance with fewer labeled instances.

In a typical AL framework (Fine et al., 2002; Freund et al., 1997; Graepel & Herbrich, 2000; Seung et al., 1992; Campbell et al., 2000; Schohn & Cohn, 2000; Tong & Koller, 2001; Ash et al., 2019b; Sener & Savarese, 2017; Wei et al., 2015; Killamsetty et al., 2020; Kirsch et al., 2019), a learner begins with a small number of labeled data points and request labels for more data points *sequentially*. For existing methods to be effective, they require multiple rounds of selection. However, the multi-round nature could be a limitation for applying AL to real-world applications, because the most common labeling platforms, e.g., Amazon Mechanical Turk and annotation outsourcing companies, usually do not support a timely interaction between the learner and data annotators. Although batch-mode active learning allows the labeling of a batch of data points in each round, it does not allow complete parallelization of labeling efforts. Another drawback of the existing AL strategies is that they rely on a small amount of labeled data points to start with. These initially labeled data are either randomly selected and sent to the annotator for labeling or assumed to already exist in the target domain. However, AL is often utilized in a new domain where no initially labeled data are available.

In this paper, we explore the possibility of zero-round AL and ask the question: *can we select unlabeled data in a way that rely on zero feedback from potential annotators but works better than random selection?* Such data selection strategies, if exist, can be directly plugged into the widely used

labeling platforms and work standalone, allowing for parallelizing labelers to reduce the overall time costs. Moreover, they can serve as a warm-start for existing multi-round AL approaches by providing better initially labeled data rather than randomly selected data. Our key idea to enable effective zero-round data selection is inspired by the observations that there are often labeled datasets available from related domains. In some applications, such as autonomous driving, there are off-the-shelf simulators that can simulate a large set of labeled data points that are related to the dataset to be labeled. Intuitively, these labeled data, although from a different domain, might still provide useful information about what types of data are worth being labeled.

In this paper, we present Domain adaptive Data Utility function Learning and Optimization ($\mathrm{D^2ULO}$), an algorithm that can leverage labeled datasets from a *source domain* to help select the unlabeled data instances in the *target domain*. Importantly, our approach does not rely on any labeled instances from the target domain and any interactions with the data labeler to design the unlabeled data selection strategy; hence, we call it a *zero-round active learning strategy*. Specifically, we train a data utility model that predicts the utility for any given unlabeled dataset, along with a feature extractor. We design our training scheme modified from domain adaptation techniques so that the feature extractor will extract domain-invariant features that are, at the same time, effective for predicting the utility of the dataset. Another important benefit enabled by the utility modeling is that our approach can provide an estimate for utility of the selected data, which is useful in practice for learners to decide the amount of unlabeled points to annotate.

Our experiments evaluate two use cases: standalone zero-round AL or warm-start for multi-round AL. We first focus on the zero-round AL setting and show that $\mathrm{D^2ULO}$ outperforms the existing state-of-the-art AL strategies equipped with domain adaptation, a natural baseline for our approach, over various domain shift settings, including the challenging one where the source domain is synthetic data while target domain is real-world data. Besides, $\mathrm{D^2ULO}$ can also be applied to the scenario where the source and target domain have different label spaces, while typical AL strategies cannot. Moreover, when used as a warm-start, our approach can effectively improve the performance of existing multi-round AL methods.

2 Related Work

Active Learning. Active learning aims to reduce labeling effort by selecting data that are most valuable for model training, and it usually performs in an iterative manner. Earlier works (Fine et al., 2002; Freund et al., 1997; Graepel & Herbrich, 2000; Seung et al., 1992; Campbell et al., 2000; Schohn & Cohn, 2000; Tong & Koller, 2001) select only one sample each round. Such AL strategies cannot parallelize labeling efforts and are often time-consuming in practice. Batch-mode active learning (Ash et al., 2019b; Sener & Savarese, 2017; Ash et al., 2019b; Wei et al., 2015; Kirsch et al., 2019; Killamsetty et al., 2020), by contrast, queries data in groups and hence improves learning efficiency; particular, it can better handle models with slow training procedures (e.g., DNNs).

AL strategies above which are designed to proceed iteratively until exceeding the labeling budget, hence require timely interaction with data labeler and is not fully parallelized. A most recent work (Wang et al., 2021) proposes DULO for one-round AL, which selects the desired amount of unlabeled points all at once based on an initially labeled set. They formulate the problem of one-round AL as the one of maximizing data utility functions, which map a dataset to some performance measure of the model trained on the set. While DULO addressed the multi-round limitation of current AL strategies, it still relies on initially labeled data to start with, which are not applicable in some scenarios. This motivates us to investigate a new setting, which AL is conducted under limited interaction with data labeler and without relying on pre-labeled data. We propose $\mathrm{D^2ULO}$ which is effective to acquire data smartly in this setting.

Domain Adaptation. Domain adaptation (DA) is a common solution to dealing with distribution shifts between source and target domain. The core idea is to learn some domain-invariant features, so the task model trained on the source domain can be readily applied to the target domain. There are three categories of DA depending on the data available from the target domain: unsupervised DA, semi-supervised DA, and supervised DA. Unsupervised DA is a setting where labeled target data is not available and agrees with the problem setting studied by our paper. Earlier works in this setting focus on minimizing some specific measurements of distributional discrepancy in the feature space. For example, Long et al. (2015), Long et al. (2017b) and Tzeng et al. (2014) characterize distribution distance via the Maximum Mean Discrepancy (MMD) of kernel embeddings; Saito et al. (2018) and Lee et al. (2019) utilizes category predictions from two task classifiers to measure the

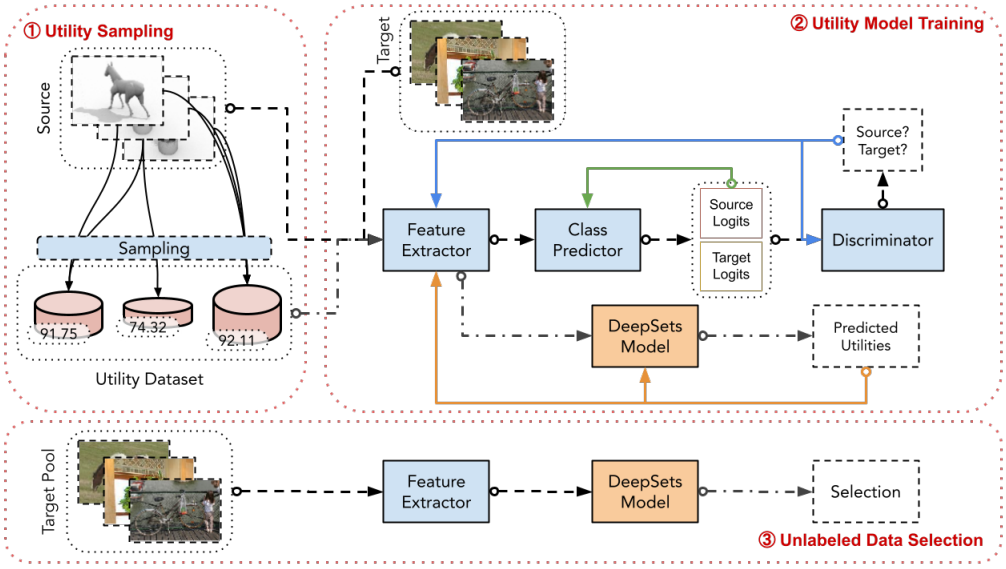

Figure 1: Overall Workflow of D^{2}ULO.

domain discrepancy. These approaches were further improved by the use of an adversarial objective loss function regarding to a domain discriminator that tries to distinguish between source and target feature embeddings (Tzeng et al., 2015; Ganin & Lempitsky, 2015; Ganin et al., 2016; Long et al., 2017a). However, the adversarial training may encounter the technical difficulty of model collapse (Mirza & Osindero, 2014). A recent work (Hoffman et al., 2018) combines generative adversarial networks (GAN) with cycle-consistent constraints and adapts representations at both feature-level and pixel-level effectively.

Combining Active Learning and Domain Adaptation. Although both active learning and domain adaptation are two possible solutions to problem of insufficient labels, only a few work in the literature integrate these two methodologies into a single framework. Chattopadhyay et al. (2013) proposes a method that re-weights source data and selects target data to query simultaneously, so that the dataset, consisting of re-weighted source samples, labeled target samples and queried target samples, is closest to the distribution of target unlabeled data. Saha et al. (2011) and Rai et al. (2010) propose ALDA that consists of three models: a domain adaptation classifier which adapts feature representation of source domain; a domain classifier that avoids querying labels for "source" data that are similar to target samples; a source classifier that provide labels for "source" data that resemble target samples. AADA, recently proposed by Su et al. (2020), starts from training a unsupervised domain adaptation classifier, then targets sample selection using importance weights. Model retraining are performed iteratively in AADA. These methods share the same limitation as most of the existing AL strategies as they are all designed to proceed for multi-rounds until exceeding the selection budget. To the best of our knowledge, D^{2}ULO is the first that leverages the idea of domain adaptation to perform AL in the zero-round setting.

3 Approach

Setting. Existing AL strategies rely on a small amount of labeled data in the target domain T. By contrast, our goal is to develop zero-round AL strategies, which do not require any labeled data in the target domain. We assume that there exists a source domain S whose distribution p_s is closely related to the target distribution p_t, while unlike target domain T, the label of instances from S is already available or easily accessible.

In this section, we introduce our algorithm D^{2}ULO. The key idea is to first learn a *data utility model* that can predict the utility for any set of unlabeled instance. We leverage domain adaptation techniques to ensure that the model is useful for predicting the utility for unlabeled instances in the target domain and further use this model to guide the data selection. We will denote labeled data by $\mathcal{L}$, unlabeled data by $\mathcal{U}$, and input and output spaces by $\mathcal{X}, \mathcal{Y}$, respectively.

Algorithm 1: Data Utility Sampling

Input : a subset of samples $\mathcal{L}^I = (\mathcal{X}^I, \mathcal{Y}^I)$ chosen from training set which the index is given by I; validation set $\mathcal{L}^{val}$; classifier f; feature extractor G_f; metric function u.

Output : utility dataset S_{DS} for DeepSets training.

Initialize utility dataset $S_{DS} = \emptyset$.
for $i = 1, \ldots, N$ **do**
    Randomly choose a subset $\mathcal{L}_i = (\mathcal{X}_i, \mathcal{Y}_i)$ where $\mathcal{L}_i \subseteq \mathcal{L}^I$.
    Train classifier f with $\mathcal{L}_i$
    $u_i \leftarrow u(f, \mathcal{L}^{val})$
    $S_{DS} = S_{DS} \cup \{(\mathcal{X}_i, u_i)\}$.
end
return S_{DS}

3.1 Overview

The concept central to our AL strategy is a *data utility function*, which maps any set of unlabeled instances to the performance the ML model trained on the set once it is labeled. With such a function, AL can be done by simply selecting the unlabeled instances that maximize the output of the data utility model. Although data utility functions may have a close form for certain types of learning algorithms and model performance metrics (e.g., the test classification accuracy of K-Nearest-Neighbor (Wei et al., 2015)), for most models data utility functions cannot analytically derived. Recent work (Wang et al., 2021) proposed to learn data utility functions from data. Note that data utility functions are set functions, in which the input is a data set and the output is a real value indicating the utility of the data. Hence, each training samples for data utility function learning consist of a set of data points and the corresponding utility score, indicating the performance of the ML model trained on the set. Constructing the training set for data utility learning could be expensive, because to label each training sample, one needs to re-train the model. Fortunately, Wang et al. (2021) presents some empirical evidence that the learning of data utility functions could be sample-efficient due to its "diminishing return" property. Also, one can replace the original ML model with a efficiently-trainable proxy model (such as logistic regression) while still retaining good data selection performance.

Note that data utility learning require labeled data instances, which make it possible to creating the training set. Wang et al. (2021) assumed a small labeled set in the target domain for data utility learning. However, this assumption no longer holds true in the zero-round AL setting. To resolve this problem, we propose to learn the data utility model on the source domain and mitigate the effects of domain shifts via domain adaptation.

3.2 D^{2}ULO Algorithm

The workflow of our algorithm is summarized in 1.

Step 1: Utility Sampling. The goal of this step is to construct the training set for learning data utility functions. Given a set of samples $\mathcal{L}^I$ and a validation set $\mathcal{L}_{val}$ in the source domain, each time we randomly sample a subset $\mathcal{L}_i \subseteq \mathcal{L}^I$ and train a classifier f on it. Utility of this subset is then given by utility metric u which in this paper is the validation accuracy of f on $\mathcal{L}_{val}$. The utility training set S_{DS} is thus $\{(\mathcal{L}_i, u_i)\}$. A general utility sampling workflow is demonstrate in Algorithm 1.

Step 2: Utility model training. The goal of this step is to train a utility model effective for predicting the utility for unlabeled data in the target domain. Following Wang et al. (2021), we adopt the popular set function model–DeepSets (Zaheer et al., 2017)–as the data utility model. DeepSets is a DNN has the property of permutation invariance and equivariance, which makes it suitable for set function modeling. Specifically, a feature extractor G_f will be utilized to get the embedding of the training instances in S_{DS}, and the DeepSets model f_{DS} maps the feature embedding of a set of points to its corresponding utility.

In the setting of interest to our paper, labeled data is not available in the target domain and the utility model f_{DS} can only be trained on data from another domain. Hence, domain adaptation is needed to mitigate the performance drop caused by domain shift.

A domain adaptation framework usually consists of three components: a feature extractor G_f, a class predictor G_y which takes the output embedding of G_f and make class predictions, and a discriminator

Algorithm 2: D^2ULO

Input : labeled source data $\mathcal{L}_s = (\mathcal{X}_s, \mathcal{Y}_s)$; unlabeled target data $\mathcal{U}_t$; utility dataset $S_{DS} = (X, U)$, where $X = (X_1, \ldots, X_N), U = (u_1, \ldots, u_N)$.
Model : $G = \{G_f, G_y, G_d\}$; feature extractor G_f; class predictor G_y; discriminator G_d; DeepSets utility model f_{DS}

```
for epoch = 1, ... do
    for k steps do
        Train G with (L_s, U_t)
    end
    Fix G_f; extract the feature embedding E_s of utility dataset E_s ← G_f(X)
    Train a DeepSets model f_DS on (E_s, U)
    Fix f_DS; train G_f with (X, U)
end
return G_f; f_DS
```

G_d that aims to distinguish between source and target domain data. DA typically has two goals: 1) map examples from two domains to a common feature space; and 2) retain useful information for classification. Those two goals are usually achieved through optimizing the GAN loss L_{GAN} and classification loss L_{cls}, given by

$$L_{GAN} = -\big[\mathbb{E}_{x\sim p_s(x)} \log G_d(G_f(x)) + \mathbb{E}_{x\sim p_t(x)} \log(1 - G_d(G_f(x)))\big]. \tag{1}$$

$$L_{cls} = \text{CrossEntropy}\left(G_y(G_f(x)), y\right) \tag{2}$$

We now discuss how to leverage domain adaptation in data utility learning to train a utility model useful for data selection in the target domain. Direct application of existing DA techniques to training the feature extractor cannot achieve a decent result on data selection, as the feature extractor learned in this way is only optimized towards the goal of being useful for classification, ignoring the goal of being useful for predicting data utility. For example, the best possible features for classification tasks would be simply the label for the data points. However, this kind of features contain no information about the quality of the data points. To solve this challenge, we modify the DA process and incorporate a DeepSets loss into the training objective in order to learn a feature representation useful for predicting data utility, which in turn facilitates data selection.

We also notice that training the data utility model together with DA models (including feature extractor, class predictor, and discriminator) simultaneously is unstable. To solve this problem, we propose an iterative training process where DeepSets model is trained and used to update the feature extractor G_f after k steps training of DA models. Such a training process realizes consistent stability in the experiments.

Specifically, given the labeled source data $\mathcal{L}_s$, unlabeled target data $\mathcal{U}_t$, a utility training set S_{DS} obtained from Algorithm 1, we alternate between k steps of general domain adaptation training and one step of utility training. The former one just follows the usual DA framework. For the latter one, a DeepSets model f_{DS} is first trained on S_{DS} given current feature extractor G_f, and it will be fixed and used to optimize G_f in turn given the same objective of minimizing DeepSets Loss:

$$\min_{f_{DS}} \min_{G_f} L_{DS} = \sum_{i=1}^{N} \|f_{DS}(G_f(X_i)) - u_i\|^2 \tag{3}$$

Note that D^2ULO can be combine with any state-of-the-art DA frameworks, and we use CyCADA (Hoffman et al., 2018), UDA (Sun et al., 2019), AFN (Xu et al., 2019) in this paper.

Step 3: Unlabeled Data Selection. The last step of D^2ULO is to seek for the unlabeled data attaining maximal utility under the learned utility model. Formally, we solve the following optimization

problem:

$$\underset{|S|=M, S\subseteq\mathcal{U}}{\arg\max} \; f_{DS}(G_f(S)) \tag{4}$$

via a stochastic greedy algorithm: for each iteration, we randomly select a subset of data and then find the best data point within that subset. When the size of subsets in the inference stage are much larger than the training size of DeepSets(like 20000 vs. 500), the generalization error could be overwhelming. In this case, one can perform block-selection proposed in (Wang et al., 2021).

4 EVALUATION

4.1 EVALUATION SETTINGS

4.1.1 EVALUATION PROTOCOL

We use two approaches to evaluate the utility of selected subset: **1) Train-from-Scratch:** we train a model from scratch on the data points selected from the target domain, and the utility of selected data points is given by the trained model's accuracy; **2) Fine-tune:** we adopt the method proposed in Zou et al. (2019) to fine-tune the classifier G_y obtained from our algorithm using both selected samples and unlabeled samples with hypothesized labels. Specifically, given a batch of labeled target samples chosen by the strategy, we compute the centroid of each class in the feature space and generate a hypothesized label for each unlabeled sample given its similarity between different centroids. We use the inverse of Wasserstein distance as the similarity metric.

.

4.1.2 BASELINE ALGORITHMS

Source	Target	Domain Adaptation
MNIST	USPS	CyCADA (Hoffman et al., 2018)
USPS	MNIST	CyCADA
SVHN	MNIST	CyCADA
CIFAR-10	STL-10	UDA (Sun et al., 2019)
VISDA-Synthetic	VISDA-Real	AFN (Xu et al., 2019)
MNIST-04	MNIST-59	N/A
MNIST-04	USPS-59	CyCADA

Table 1: Dataset and Training Settings.

For baseline algorithms, we combine state-of-the-art active learning strategies with domain adaptation. Specifically, we pre-train a feature extractor that minimizes the distance between source and target domain in the feature space, apply it to extract features for the unlabeled data pool and perform active data selection on the extracted features. Note that most of these existing AL strategies cannot be directly applicable to the zero-round AL setting.

We compare D^2ULO with the following state-of-the-art batch active learning strategies equipped with domain adaption. Specifically, our baselines contain as follows:

- **FASS.** Wei et al. (2015) performs subset selection as maximization of Nearest Neighbor submodular function on unlabeled data with hypothesized labels.
- **BADGE.** Ash et al. (2019a) selects a subset of samples with hypothsized label whose gradients span a diverse set of directions.
- **GLISTER.** Killamsetty et al. (2020) formulates the selection as a discrete bi-level optimization on samples with hypothesized labels.
- **AADA.** Su et al. (2020) uses a sample selection criterion which is the product of importance estimation and entropy of unlabeled data.
- **Random.** In this setting we randomly select a subset from all the unlabeled target data.

Moreover, we also train an "optimal" DeepSets model on labeled *target* domain data, which corresponds to DULO, serves as an upper bound of the active learning performance with only labeled source domain data available. We label this upper bound with **Optimal.** Note that this upper bound is *not* realizable in the zero-round AL setting because of the lack of labeled target domain data. We plot this setting in the figures only to better understand how much room our strategy could be further improved.

4.1.3 DATASETS AND IMPLEMENTATION DETAILS

Table 1 summarizes the datasets and implementation settings. We evaluate the performance of D^{2}ULO and baseline approaches over four pairs of domain shifts: MNIST $\Rightarrow$ USPS, USPS $\Rightarrow$ MNIST, SVHN $\Rightarrow$ MNIST, CIFAR10 $\Rightarrow$ STL10. We also evaluate two more challenging transfer settings, where the source domain has inconsistent labels with the target domain: MNIST with digits 0-4 $\Rightarrow$ MNIST with digits 5-9, as well as MNIST with digits 0-4 $\Rightarrow$ USPS with digits 5-9. None of the baselines is applicable to these two settings by design. Following the settings in prior work (Killamsetty et al., 2020; Wang et al., 2021), we examine the effectiveness of different strategies on robust data selection, where partial data is corrupted by white noise to simulate real-world scenario. Specifically, we add Gaussian noise with different scales to 85% unlabeled data of USPS, 75% unlabeled data of MNIST, 25% of STL-10 and 20% of VISDA-Real.

For all the source datasets, we randomly sample 300 (MNIST, USPS) or 500 (SVHN, CIFAR10, STL10) data points of the training set as $\mathcal{L}^I$ to perform data utility sampling demonstrated in Algorithm 1. We follow the implementation of DULO (Wang et al., 2021) to set $N = 5000$ and split the obtained S_{DS} into training and validation set with a ratio $4:1$. We use small models (i.e., SVM, Logistic, Small CNN) as the classifier f in Algorithm 1 to obtain the data utility. This is because f needs to be trained for thousands times to construct an utility dataset. DULO (Wang et al., 2021) empirically finds that data utility functions for small models are positively correlated with those for large models. Since data selection based on utility models only relies on the relative utility values between different set, utility models trained on samples obtained from small proxy models could still be useful for selecting data for large models.

We consider the state-of-art domain adaptation techniques for the specific transfer settings. Specifically, we combine our method with three different DA frameworks: CyCADA (Hoffman et al., 2018), UDA (Sun et al., 2019), and AFN(Xu et al., 2019), and the DA framework used for each transfer setting is given in Table 1. For training the DeepSets model, we use the same hyper-parameter as Wang et al. (2021): we use Adam optimizer with learning rate $1e-5$, mini-batch size of 32, β_1 = 0.9, and $\beta 2$ = 0.999.

For Fine-tune performance evaluation, the starter model is G_f in the corresponding domain adaptation framework in each setting. For Train-from-Sratch evaluation, we use three types of models to calculate the performance: 1) SVM, which is implemented with scikit-learn (Pedregosa et al., 2011) with regularization parameter $C = 0.1$; 2) Logistic model; and 3)Small CNN model which has two convolutional layers and two max pooling layers and three fully-connected layers. Adam optimizer with learning rate $1e-3$, $\varepsilon = 1e-7$ is used for training the small CNN model.

We use GeForce RTX 2080 ti for experiments on VISDA and NVIDIA Tesla K80 GPU for all the other experiments.

4.2 EXPERIMENT RESULTS

Real-to-Real Adaptation. We start from comparing D^{2}ULO with baselines on various domain shifts between real datasets.

Figure 2 shows the results averaged over multiple random seeds. The x axis shows the number of target sample selected by different strategies, and the y axis is the accuracy of the model trained on selected points. In this figure, D^{2}ULO outperforms all baselines in various shifts. The accuracy drop of fine-tune in Figure 2 (a) is caused by the large portion of noise in the unlabeled data pool. After the clean data are all selected (300 clean data points in total), more and more noisy data will be picked as the selection process proceeds, which degrades the model performance. Note that while all the baselines exhibit poor performance from beginning to end, our method appears to improve the accuracy significantly before 600 data points are selected (at least 300 noisy samples are included), indicating our superior ability to recognize noisy samples. Another interesting finding is that the margin between D^{2}ULO and Optimal

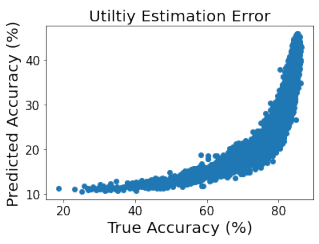

Figure 3: True Utiltiy vs Estimated Utility. (Spearman's correlation coefficient is 0.96.)

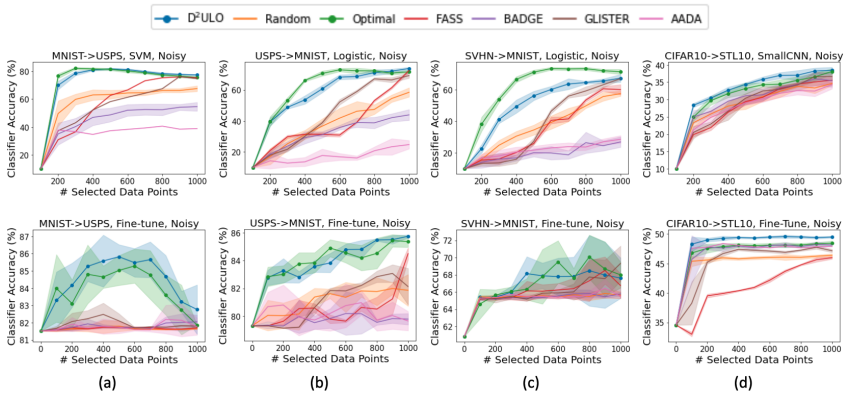

Figure 2: Performance of D^2ULO on various adaptation shifts. The first row gives the results of Train-from-Scratch, where 'SVM', 'Logistic' and 'SmallCNN' indicate the model used for obtaining utilities. The second row give the results of Fine-tune, and the start points are the classifier accuracy on target validation set after domain adaptation.

is small, except for Figure 2 (d) where Optimal is worse than D^2ULO. This may caused by the overfitting of DeepSets.

Another advantage of D^2ULO over existing AL strategies is that we can provide a utility estimate for the selected data using the data utility model. Such a utility estimate could be very useful in practice for making an informed decision about the labeling budget. We compare our DeepSets estimated utility with the true utility for the MNIST $\Rightarrow$ USPS setting. Here, we randomly choose 300 data points of the target domain (USPS) and perform Algorithm 1 to sample 4000 subsets. The true utilities is given by a SVM model trained on the subsets. Note that these 300 data points are unseen during DeepSets training. Figure 3 shows that the estimated utility underestimates the true utility systematically but is substantially positively correlated with the true utility, with a Spearman's correlation coefficient of 0.96. Such underestimating may be due to the target domain (USPS) being easier to learn than the source domain (MNIST). Hence, the utility estimates provided by D^2ULO can serve as a lower bound on the actual utility, which is still useful for guiding the choice of labeling budget. With better modeling between the relationship between the estimated and the true utility, one may be able to correct the bias in our estimation. We leave the exploration of this interesting direction to future work. Figure 3 also sheds light on the strong ability of D^2ULO on differentiating unlabeled data quality in the target domain, even without the access to any labeled data from the domain.

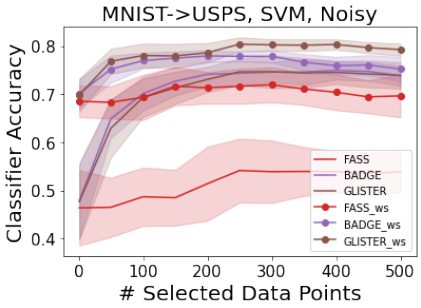

Figure 4: Performance of existing AL approaches with 100 initially labeled data points selected randomly or by D^2ULO (warm start, denoted by '_ws').

Synthetic-to-Real Adaptation. We further study the effectiveness of different strategies on the synthetic-to-real transfer setting. This setting could have great practical value because in many application domains, there exist sophisticated simulators that can generate a large amount of labeled data. We experiment on the VISDA-2017 dataset which has significant synthetic-to-real domain gap. The source domain of VISDA are synthetic images generated by rendering from 3D models; the target domain are real object images collected from Microsoft COCO (Lin et al., 2014) and contains some natural variations in image quality.

As shown in Figure 5, D^2ULO achieves the best performance among all the strategies. We also notice that even a very small amount of labeled target data can help improve the classifier accuracy by a large margin. For instance, in Figure 5 (b), the Fine-tune accuracy increases rapidly at the beginning while only 100 data points are selected. This emphasizes the need of selecting data in the

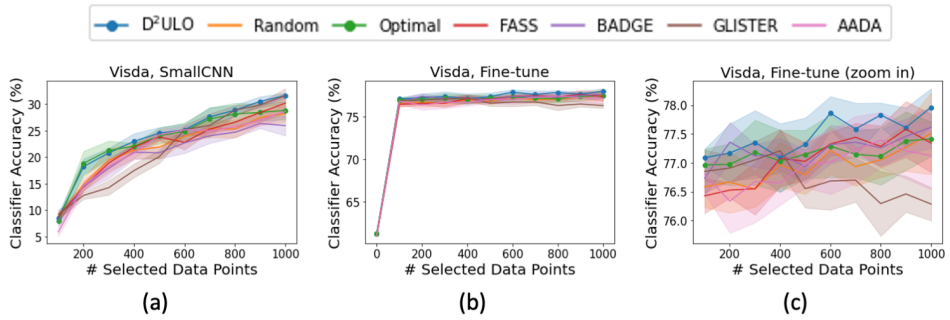

Figure 5: VISDA-2017 result: synthetic ⇒ real. (a) gives the results of Train-from-Scratch, (b) and (c) give the result of Fine-tune. One interesting finding is that, all the strategies achieve a large performance improvement on classification accuracy. This indicates the needs to select data points from the target domain.

target domain for further improvement of domain adaptation performance. Since the target domain is relatively clean, random baseline works already very well.

Label Mismatch. There are many real-world datasets that do not have overlap in the label space or only share a few common classes. Hence, we also conduct experiments in the setting where the source domain has entirely different object categories from the target domain. Specifically, we use digit 0-4 of MNIST dataset as the source domain, digit 5-9 of MNIST and USPS datasets as the target domain. This setting has great practical value yet has not been studied by previous AL literature. The reason is that most AL strategies rely on hypothesized labels generated by the classifier trained on the source domain and they become infeasible in this setting. For the same reason, we omit the Fine-tune performance metric and only report the accuracy of Train-from-Scratch. As we can see from Figure 6, D^2ULO outperforms random by a large margin and is comparable to Optimal.

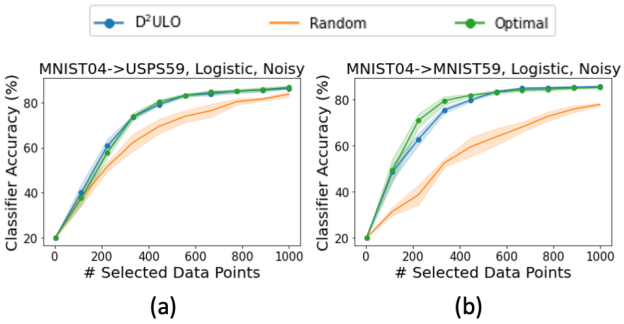

Figure 6: Performance of D^2ULO on domains that have inconsistent label space.

Warm-Starting Existing Multi-Round Approaches. As shown in Figure 4, using D^2ULO as a warm start, the performance of existing AL approaches can be consistently improved by a large margin.

5 Future Work

There are many interesting venues for future work. For instance, in our experiments, we observe that the DeepSet-based utility learning often overfits to the training samples, which directly affects the efficacy of subsequent data selection tasks. One interesting future work is to develop preventative measure against overfitting for DeepSets training via new training algorithm, model architecture, and regularization techniques. It is also interesting to explore the application to domains beyond image and study how to customize the synthetic data generation to the goal of improving active learning performance.

REFERENCES

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

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*, 2019a.

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*, 2019b.

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

Rita Chattopadhyay, Wei Fan, Ian Davidson, Sethuraman Panchanathan, and Jieping Ye. Joint transfer and batch-mode active learning. In *International conference on machine learning*, pp. 253–261. PMLR, 2013.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

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

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.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pp. 1180–1189. PMLR, 2015.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

Thore Graepel and Ralf Herbrich. The kernel gibbs sampler. In *NIPS*, pp. 514–520. Citeseer, 2000.

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pp. 1989–1998. PMLR, 2018.

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.

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.

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

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

Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10285–10295, 2019.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pp. 97–105. PMLR, 2015.

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. *arXiv preprint arXiv:1705.10667*, 2017a.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pp. 2208–2217. PMLR, 2017b.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

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.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

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.

Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge, 2017.

Piyush Rai, Avishek Saha, Hal Daumé III, and Suresh Venkatasubramanian. Domain adaptation meets active learning. In *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*, pp. 27–32, 2010.

Avishek Saha, Piyush Rai, Hal Daumé, Suresh Venkatasubramanian, and Scott L DuVall. Active supervised domain adaptation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 97–112. Springer, 2011.

Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3723–3732, 2018.

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

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

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.

Jong-Chyi Su, Yi-Hsuan Tsai, Kihyuk Sohn, Buyu Liu, Subhransu Maji, and Manmohan Chandraker. Active adversarial domain adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 739–748, 2020.

Yu Sun, Eric Tzeng, Trevor Darrell, and Alexei A. Efros. Unsupervised domain adaptation through self-supervision, 2019.

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.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE international conference on computer vision*, pp. 4068–4076, 2015.

Tianhao Wang, Si Chen, and Ruoxi Jia. One-round active learning. *arXiv preprint arXiv:2104.11843*, 2021.

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

Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

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

Han Zou, Yuxun Zhou, Jianfei Yang, Huihan Liu, Hari Prasanna Das, and Costas J Spanos. Consensus adversarial domain adaptation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 5997–6004, 2019.

A APPENDIX

A.1 DETAILS OF DATASETS USED IN SECTION 4

MNIST (LeCun, 1998). MNIST dataset contains a training set of 60,000 examples and a test set of 10,000 examples. The images are grayscale handwritten digits with size 28×28. We resize the images to 32×32 in setting SVHN $\Rightarrow$ MNSIT.

USPS (Alpaydin & Kaynak, 1998). USPS dataset is a digit dataset scanned from envelopes. It contains a total of 9,298 16×16 grayscale pixels. We resize them to 28×28 in both MNIST $\Rightarrow$ USPS and USPS $\Rightarrow$ MNIST setting.

SVHN (Netzer et al., 2011). SVHN is a real-world color house-number dataset containing 73,257 images for training and 26,032 images for testing. We use the version where all digits have been resized to 32×32 pixels.

CIFAR-10 (Krizhevsky et al., 2009). The CIFAR-10 is an image recognition dataset containing 60,000 32×32 3-channel images in 10 classes.

STL-10 (Coates et al., 2011). The STL-10 dataset consists of 13,000 color images of size 96×96 in 10 classes. We resize them to 32×32 in the experiments.

VISDA2017 (Peng et al., 2017). VISDA2017 dataset is designed for unsupervised domain adaptation challenge which contains more than 280K images across 12 object categories with large domain gap. The source domain are synthetic 2D images rendering of 3D models which the angles and lighting conditions are different. The target domain are photo-realistic or real-images. In the experiment, we resize all the images to 256×256 and crop at the center obtaining images with size 224×224. An example of synthetic-real image pair is shown in Figure 7.

Figure 7: Example images in VISDA2017. The left is an image of source domain (synthetic) while the right is an image of target domain (real).

A.2 DETAILS OF MODELS AND BASELINE ALGORITHMS IN SECTION 4

SVM. We use Linear Support Vector Classification (SVC) implemented by scikit-learn (Pedregosa et al., 2011) with L2 penalty and regularization parameter $C = 0.1$. Others remain as default.

Logistic Regression. We use Logistic Regression implemented by scikit-learn (Pedregosa et al., 2011). We set the maximum number of iterations to be 1000.

Small CNN. The small CNN model we used has two convolutional layers and two max pooling layers and three fully-connected layers. We use Adam optimizer with learning rate 10^{-3}, $\varepsilon = 10^{-7}$, batch size 32 for training the small CNN model.

DeepSets Model. A DeepSets model can be represented as $f_{DS}(S) = \rho(\sum_{x \in S} \phi(x))$ where both ρ and ϕ are neural networks. In our experiments, both ρ and ϕ contain 3 linear layer with ELU activation, and we set the number of neurons to be 256 in each hidden layer, the dimension of set features which is the output of ϕ network to be 256. For training DeepSets models, we use Adam optimizer with learning rate 10^{-5}, batch size 32, $\beta_1 = 0.9$, and $\beta_2 = 0.99$.

Baseline AL Techniques. We use BADGE, FASS, and GLISTER implemented by DISTIL[1]. Specifically, we set batch size to be 32 for all of the three strategies, and learning rate to be 0.001 for glister.

A.3 Other Implementation Details

Domain Adaptation. We test our method with three state-of-the-art domain adaptation frameworks in this paper: CyCADA (Hoffman et al., 2018), UDA (Sun et al., 2019), AFN (Xu et al., 2019).

For CyCADA[2], we follow their official implementation where a source classifier is firstly trained using Adam optimizer with learning rate 10^{-4}, batch size 128, $\beta_1 = 0.9$, and $\beta_2 = 0.99$. Then, weights of this source classifier are used as the initial weights of target classifier to perform domain adaptation. Same optimizer are used for training target classifier. We set the k in Line 10 of Algorithm 2 to be 10.

For UDA[3], we use SGD optimizer with initial learning rate 0.1. We later decay the learning rate to 0.001 after 10 epochs. And we set k to be 5.

For AFN[4], we use SGD optimizer with learning rate 0.001 and weight decay 5×10^{-4} for training feature extractor, and SGD optimizer with learning rate 0.001, momentum 0.9 and weight decay 5×10^{-4} for training class predictor. We set k to be 5.

When integrating all of the above three DA frameworks into D^{2}ULO, we use the same Adam optimizer with learning rate 10^{-6}, $\beta_1 = 0.9$, and $\beta_2 = 0.99$ for DeepSets Loss back-propagation.

Data Selection. We apply stochastic greedy optimization (Mirzasoleiman et al., 2015) to solve Equation (4), and we set $\epsilon = 10^{-3}$.

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

[2] https://github.com/jhoffman/cycada_release

[3] https://github.com/yueatsprograms/uda_release

[4] https://github.com/jihanyang/AFN