Composite Active Learning: Towards Multi-Domain Active Learning with Theoretical Guarantees

Guang-Yuan Hao[1 4], **Hengguan Huang**[3], **Haotian Wang**[5] **Jie Gao**[2] **Hao Wang**[2]

[1]Hong Kong University of Science and Technology [2]Rutgers University
[3]National University of Singapore [4]Mohamed bin Zayed University of Artificial Intelligence [5]JD Logistics
guangyuanhao@outlook.com, hengguan.huang@gmail.com, wanghaotian18@jd.com, {jg1555, hw488}@rutgers.edu

Abstract

Active learning (AL) aims to improve model performance within a fixed labeling budget by choosing the most informative data points to label. Existing AL focuses on the single-domain setting, where all data come from the same domain (e.g., the same dataset). However, many real-world tasks often involve multiple domains. For example, in visual recognition, it is often desirable to train an image classifier that works across different environments (e.g., different backgrounds), where images from each environment constitute one domain. Such a multi-domain AL setting is challenging for prior methods because they (1) ignore the similarity among different domains when assigning labeling budget and (2) fail to handle distribution shift of data across different domains. In this paper, we propose the first general method, dubbed composite active learning (CAL), for multi-domain AL. Our approach explicitly considers the domain-level and instance-level information in the problem; CAL first assigns domain-level budgets according to domain-level importance, which is estimated by optimizing an upper error bound that we develop; with the domain-level budgets, CAL then leverages a certain instance-level query strategy to select samples to label from each domain. Our theoretical analysis shows that our method achieves a better error bound compared to current AL methods. Our empirical results demonstrate that our approach significantly outperforms the state-of-the-art AL methods on both synthetic and real-world multi-domain datasets.

1 Introduction

The performance of machine learning models, especially supervised learning ones, largely hinges on the amount of labeled data available. However, in practice, labels are often expensive, tedious, or time-consuming to obtain. Active learning (AL) tackles this problem by 'actively' choosing the most informative data points (e.g., data with the most uncertain model predictions) to label, thereby achieving higher accuracy with the same labeling budget.

Existing AL research mostly focuses on the single-domain setting, where all data come from the same domain, e.g., the same dataset. In practice, however, real-world tasks often require actively querying labels among multiple domains with the same input space. For example, to train an object recognition model that detects and classifies wildlife animals in different environments, where images from each environment constitute one domain, one needs to carefully decide how to spend the labeling budget among the different domains to achieve the highest average accuracy across all domains. In such cases, it is sub-optimal to directly perform single-domain AL separately for each domain. The reasons are two-fold. (1) *Domain similarity*: Single-domain AL fails to effectively consider the similarities of different domains when considering cross-domain generalization and assigning labeling budget. In the wildlife detection example, it would be more cost-effective to spend more computing resources and more budget on representative environments, i.e., domains that are more similar to other domains, as the improvement on such domains could better generalize to other domains, as shown in Fig. 1. (2) *Distribution shift*: Single-domain AL fails to handle the distribution shift of data across different domains because it tends to learn domain-specific features instead of domain-invariant features, leading to poor cross-domain generalization and rendering the query strategies less effective.

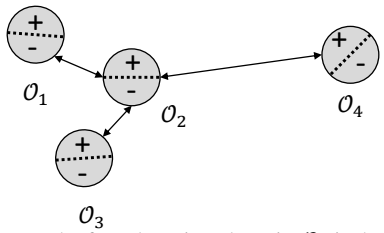

Figure 1: Among the four domains, domain $\mathcal{O}_2$ is the closest to the other three domains. Assigning more labels to this representative domain $\mathcal{O}_2$ could better generalize to other domains, since similar domains may have similar decision boundaries.

Therefore the key challenges for effective multi-domain AL are to estimate/incorporate domain similarity and to handle distribution shift. Our approach for addressing both challenges starts by constructing a *surrogate domain* for each original domain. Specifically, a surrogate domain consists of a weighted collection of all labeled data points from all domains, where the weights reflect the similarity among different domains. For example, the weight between two similar domains is expected to be higher than that between two distant domains. To estimate and incorporate domain similarity, we develop an upper bound on the average error of all domains and then estimate the similarity weights among the domains by minimizing this bound. To handle distribu-

tion shift, we use the same upper bound as the loss function to learn an encoder that maps input data from different domains into an aligned feature space, thereby reducing the distribution shift.

Specifically, our theoretical analysis shows that: (1) by jointly estimating the similarity weights and learning the encoder, one can minimize the upper bound for the average error among all domains; (2) the optimal strategy is achieved when each domain's labeling budget is proportional to its total similarity weight, i.e., the sum of similarity weights from one domain to all N domains. We summarize our contributions as follows:

- We propose Composite Active Learning (CAL) as the first general deep AL method to take into consideration both domain-level and instance-level information for addressing the problem of multi-domain active learning.
- We analyze our method and provide theoretical guarantees that CAL with our budget assignment strategy achieves a better upper bound on the average error of all domains.
- We provide empirical results on both synthetic and real-world datasets with detailed ablation studies, showing that CAL significantly improves performance over the state of the art for multi-domain active learning.

2 Related Work

Active Learning. There is rich literature on active learning [31, 36, 3, 15, 19, 4, 7]. Typically they apply a query strategy to find the most informative unlabeled samples to label, thereby achieving improved accuracy given a fixed labeling budget. Common query strategies include uncertainty-based strategies to choose data with high uncertainty [32, 27, 24, 23, 42, 34, 40, 11], density-based methods to choose representative samples [35, 38, 17, 18, 10, 14], and hybrid approaches to balance uncertainty and diversity of chosen samples [2, 8, 5, 21, 22]. A few early works are related to both active learning and multi-domain learning for specific applications such as text classification [25] and recommend system [47]. However, they are limited to linear methods and need careful feature engineering; therefore they are not applicable to our general setting that often involves highly nonlinear data and deep learning (see Sec. 3 in the Supplement for detailed discussion and results). Different from these previous works, our method focuses on the general multi-domain active learning setting. We also note that our method does not assume specific *instance-level* query strategies and is therefore *compatible with (and orthogonal to) any previous AL methods (that is, our proposed framework can be used to extend any single-domain AL methods to the multi-domain setting)*, as shown in later sections.

Active Domain Adaptation. A few recent works on domain adaptation [45, 37, 26, 46, 43, 6], a different problem setting, utilize AL to improve performance in a target domain [33, 39, 28, 13]. Their key ideas are to first perform domain adaptation to match the distributions of the source and target domains, and then apply query strategies to select useful unlabeled samples from the target domain. Here we note several key differences between active domain adaptation and multi-domain AL. (1) Active domain adaptation distinguishes between source domains and target domains, while multi-domain AL does not. (2) Active domain adaptation assumes access to all labels in the source domains even from the beginning, while multi-domain AL starts with only unlabeled data in all domains (except that in AL's initial round, one would randomly sample a few data points to label). (3) Active domain adaptation aims to improve performance (e.g., accuracy) only on the target domain, while multi-domain AL aims to improve the average performance of all domains. Such differences preclude its direct application to our multi-domain active learning settings; in Sec. 5, we show that even after careful adaptation to our setting, active domain adaptation often underperform even naive active learning baselines.

3 Methodology

In this section, we review the basics of single-domain AL, and formalize the **MU**lti-**D**omain **A**ctive **L**earning (**MUDAL**) setting. We then revisit our key ideas in Sec. 1 and describe our methods.

3.1 Preliminaries: Single-Domain Active Learning

Existing AL methods typically consider a single-domain setting and aim to optimize performance by using a *query strategy* to choose data points to label, within a given budget. Specifically, an AL model is trained in $R+1$ rounds with a total labeling budget of $M = m_0 + R \times m$. In this process, m_0 data points, randomly sampled from the unlabeled training set, are labeled in the initial round. In each of the following R rounds during the query stage, the encoder extracts features from all unlabeled data. A query strategy then utilizes these features to select and label additional m samples.

3.2 Multi-Domain Active Learning

Notation. With the input and labels denoted as $\mathbf{x}$ and y, respectively, we denote the N *original domains*' associated input data distributions as $\{\mathcal{O}_i(X)\}_{i=1}^N$ and the corresponding joint input-label distributions as $\{\mathcal{O}_i(X,Y)\}_{i=1}^N$; all domains share the same input space and label space. We assume a shared encoder e and a shared classifier h for all domains. We denote as $\mathbf{z} = e(\mathbf{x})$ the feature extracted by the encoder and $\{\mathcal{O}_i(Z)\}_{i=1}^N$ the N domains' feature distributions. Accordingly, we use X, Y, and Z to denote random variables. We treat the already labeled data points as separate domains (more details in Sec. 3.3) and therefore have N additional *labeled domains* with feature distributions $\{\mathcal{L}_j(Z)\}_{j=1}^N$. Similarly we have N additional *surrogate domains* $\{\mathcal{S}_i(Z)\}_{i=1}^N$, each of which is a mixture of the N labeled domains. With slight notation overload, we use $\mathcal{O}_i$, $\mathcal{S}_i$, $\mathcal{L}_j$ to denote original, surrogate, and labeled domains, respectively.

Problem Setting. We assume a two-step procedure for MUDAL. Specifically, with $R+1$ training rounds, denote $M = m_0 + R \times m$ as the total budget, where $m = \sum_j m_j^{(r)}$ ($m_j^{(r)}$ denotes the sub-budget for domain j in round r). In

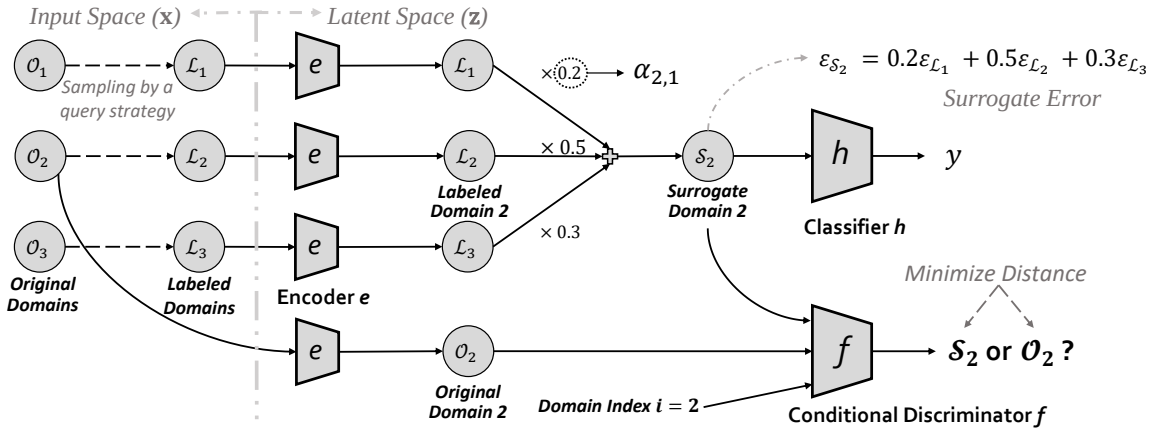

Figure 2: **Overview of our proposed framework. Left:** 3 labeled domains and 3 original domains in the input space. **Right:** In the latent space, CAL constructs surrogate domain $\mathcal{S}_2$ using α-weighted labeled domains $\mathcal{S}_2(Z) = \sum_{j=1}^{3} \alpha_{2,j}\mathcal{L}_j(Z)$, and estimate similarity weights $\alpha_{2,j}$ by minimizing the distance between surrogate domain $\mathcal{S}_2$ and its original domain $\mathcal{O}_2$. All encoders e share the same parameters. The encoder e, domain similarity $\{\alpha_{2,j}|j = 1, 2, 3\}$, and conditional discriminator f play a min-max game to reduce the distance between $\mathcal{S}_2$ and $\mathcal{O}_2$ by joint similarity estimation and feature alignment. The classifier h is trained on all surrogate domains $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$. For clarity, we omit surrogate domains $\mathcal{S}_1$ and $\mathcal{S}_3$.

Round 0, $\frac{m_0}{N}$ instances are randomly sampled from each domain as the initial training set. Each remaining round r consists of two steps: (1) *domain-level selection*, where a MUDAL algorithm decides on an additional sub-budget $m_j^{(r)}$ for each domain j, and (2) *instance-level selection*, where the algorithm applies a query strategy to choose $m_j^{(r)}$ data points in each domain j to label. The goal is to estimate the optimal sub-budget $m_j^{(r)}$ such that the average classification error among all domains can be minimized. To facilitate analysis below, we further define $\beta_j^{(r)}$ as domain j's proportion of the total budget until round r, i.e., $\beta_j^{(r)} = \frac{m_0/N + \sum_{k=1}^{r} m_j^{(k)}}{m_0 + r \times m}$.

3.3 Method Overview

Below we provide an overview for a *simplified* version of CAL including domain-level and instance-level selection, using Fig. 2 as a running example.

Constructing Surrogate Domains. A labeled domain cannot fully represent its original domain; this is because labeled data are not I.I.D. samples from the original domain due to query strategies. Therefore in each round r, our approach starts by constructing a *surrogate domain* for each original domain. Specifically, a surrogate domain consists of a weighted collection of all labeled domains (defined in Sec. 3.2), where the weights reflect the similarity among different domains. Fig. 2(left) shows an example with $N = 3$ domains in the input space. Correspondingly we have 3 labeled domains $\{\mathcal{L}_j\}_{j=1}^{3}$, where data are already sampled by query strategies and labeled in the previous rounds during AL, and 3 original domains $\{\mathcal{O}_i\}_{i=1}^{3}$, which include both labeled and unlabeled data (i.e., the size of an original domain remains constant across rounds). For each original domain $\mathcal{O}_i(Z)$ in the latent space induced by the encoder e, e.g., $\mathcal{O}_2(Z)$ at the bottom of Fig. 2(right), we construct a surrogate domain $\mathcal{S}_i(Z)$ as a mixture of all labeled domains, i.e., $\mathcal{S}_i(Z) = \sum_{j=1}^{3} \alpha_{i,j}\mathcal{L}_j(Z)$, where $\alpha_{i,j} \geq 0$ and $\sum_j \alpha_{i,j} = 1$. Fig. 2(right) shows an example of constructing surrogate domain $\mathcal{S}_2$ for original domain $\mathcal{O}_2(Z)$ as $\mathcal{S}_2(Z) = 0.2\mathcal{L}_1(Z) + 0.5\mathcal{L}_2(Z) + 0.3\mathcal{L}_3(Z)$. Accordingly, with classifier h taking $\mathcal{S}_2(Z)$ as input, $\epsilon_{\mathcal{S}_2} = 0.2\epsilon_{\mathcal{L}_1} + 0.5\epsilon_{\mathcal{L}_2} + 0.3\epsilon_{\mathcal{L}_3}$ is the surrogate error to approximate original domain $\mathcal{O}_2$'s error.

Domain Similarity (Importance) Estimation. Each surrogate domain $\mathcal{S}_i$ is a similarity-weighted sum of the N labeled domains $\{\mathcal{L}_j\}_{j=1}^{N}$. Given an encoder e, we can then estimate the similarity weight $\alpha_{i,j}$ between labeled domain $\mathcal{L}_j$ and original domain $\mathcal{O}_i$, by minimizing the distance between the original domain feature distribution $\mathcal{O}_i(Z)$ and the surrogate domain feature distribution $\mathcal{S}_i(Z)$, denoted as $d(\mathcal{O}_i(Z), \mathcal{S}_i(Z))$. For example, in Fig. 2(right) we estimate the similarity weights $\{\alpha_{2,j}\}_{j=1}^{3}$ by minimizing the distance $d(\mathcal{O}_2(Z), \mathcal{S}_2(Z))$, where $\mathcal{S}_2(Z) = \sum_{j=1}^{3} \alpha_{2,j}\mathcal{L}_j(Z)$.

Distribution-Shift Reduction with Feature Alignment. To reduce distribution shift across domains, we propose to learn an encoder e such that feature distributions from different domains can be aligned. Suppose the similarity weights $\{\alpha_{i,j}\}$ are given, we search for an encoder e that minimizes the distance $d(\mathcal{O}_i(Z), \mathcal{S}_i(Z)) = d(\mathcal{O}_i(e(X)), \mathcal{S}_i(e(X)))$. Note that this is different from adversarial domain adaptation [16] which directly aligns different $\mathcal{O}_i(Z)$'s; our preliminary results show that such direct alignment does not improve, and sometimes even hurts performance.

Joint Similarity Estimation and Feature Alignment. Finally, we perform joint optimization combining both similarity estimation (which learns $\alpha_{i,j}$) and feature alignment (which learns the encoder e). As shown in Fig. 2, The encoder e, similarity weights, and conditional discriminator f play a min-max game to reduce the distance between surrogate domain $\mathcal{S}_2$ and its original domain $\mathcal{O}_2$ in the latent (feature) space with joint optimization. Further theoretical analysis shows that such joint optimization provides a tighter upper bound for the average classification error of all domains (more details in Sec. 3.4 and Sec. 4).

Domain-Level Selection as Sub-Budget Decision Using Similarity Weights. With the estimated similarity $\alpha_{i,j}$, we then assign budget such that representative domains, i.e., do-

mains that are more similar (important) to other domains (larger $\sum_{i=1}^{N} \alpha_{i,j}$), will receive more labeling budget.

Instance-Level Selection. With the assigned domain-level budget, we then apply an instance-level query strategy (e.g., uncertainty-based strategy [31]) to select data points to label.

Remark: Domain-Level Selection Is Agnostic to Instance-Level Selection. Domain-level selection is agnostic to, and therefore compatible with, any instance-level selection methods. We consider such independence as an advantage of our CAL framework; any existing instance-level selection method, such as Margin [32] and BADGE [2], can be used as a sub-routine for CAL to work with our proposed domain-level selection algorithm. Therefore, CAL's key innovation is in (1) the first general two-level framework for multi-domain AL and (2) the first general domain-level selection method, rather than inventing new instance-level selection methods.

3.4 Composite Active Learning

Upper Bound for the Average Error of All Domains. We are now ready to present our full method, dubbed Composite Active Learning (CAL), to jointly estimate similarity weights, perform feature alignment, and assign sub-budget across domains. Formally, CAL tries to minimize an upper bound for the average error of all original domains $\frac{1}{N}\sum_i \epsilon_{\mathcal{O}_i}(h \circ e)$.

Our bound consists of three terms: **(i)** an α_j-weighted average of errors in all labeled domains $\sum_j \alpha_j \epsilon_{\mathcal{L}_j}(h \circ e)$, where $\alpha_j = \frac{1}{N}\sum_i \alpha_{i,j}$ is the summary similarity score for labeled domain $\mathcal{L}_j$, $h(\cdot)$ is the classifier, and $\epsilon_{\mathcal{L}_j}(h \circ e)$ is the expected classification error in labeled domain $\mathcal{L}_j$ given $h(\cdot)$ and $e(\cdot)$; **(ii)** the average $\mathcal{H}\Delta\mathcal{H}$-distance [6] between each original domain $\mathcal{O}_i$ and its surrogate domain $\mathcal{S}_i$, denoted as $\frac{1}{N}\sum_i \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{O}_i(e(X)), \mathcal{S}_i(e(X)))$; **(iii)** an h-agnostic term $\frac{1}{N}\lambda_i$, where $\lambda_i = \min_{h_i}(\epsilon_{\mathcal{O}_i}(h_i \circ e) + \epsilon_{\mathcal{S}_i}(h_i \circ e))$ and h_i is the i-th domain-specific classifier; $\epsilon_{\mathcal{O}_i}(h_i \circ e)$ and $\epsilon_{\mathcal{S}_i}(h_i \circ e))$ are the expected classification error in original domain $\mathcal{O}_i$ and surrogate domain $\mathcal{S}_i$, respectively. We summarize the upper bound below (proofs in Sec. 4 and Sec. 2 of the Supplement):

$$\min_{h,e} \min_{\boldsymbol{\alpha}} \sum_j \alpha_j \epsilon_{\mathcal{L}_j}(h \circ e) + \frac{1}{N}\sum_i \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{O}_i(e(X)), \mathcal{S}_i(e(X))) + \frac{1}{N}\sum_i \lambda_i, \quad (1)$$

where i indexes original domains and surrogate domains while j indexes labeled domains. $\boldsymbol{\alpha} = [\alpha_{i,j}]_{i=1,j=1}^{N,N}$ is a similarity matrix containing the similarity weights. Note that by the definition of surrogate domains, the first term $\sum_j \alpha_j \epsilon_{\mathcal{L}_j}(h \circ e) = \frac{1}{N}\sum_i \epsilon_{\mathcal{S}_i}(h \circ e)$.

The surrogate $\mathcal{S}_i(e(X)) = \sum_j \alpha_{i,j} \mathcal{L}_j(e(X))$ is a weighted average of all labeled domains; therefore $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{O}_i(e(X)), \mathcal{S}_i(e(X)))$ *connects the labeled domain error, which can be computed, to the original domain error, which we want to minimize, through our constructed surrogate domains* $\mathcal{S}_i$.

Objective Function. The three terms in Eqn. 1 lead to a minimax game as our objective function.

The first term $\sum_j \alpha_j \epsilon_{\mathcal{L}_j}(h \circ e)$, equal to $\frac{1}{N}\sum_i \epsilon_{\mathcal{S}_i}(h \circ e)$, in Eqn. 1 leads to the first term of our objective function:

$$V_h(h, e, \boldsymbol{\alpha}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}^{\mathcal{S}_i}[L_Y(h \circ e(\mathbf{x}), y)] = \sum_{j=1}^{N} \alpha_j \mathbb{E}^{\mathcal{L}_j}[L_Y(h \circ e(\mathbf{x}), y)], \quad (2)$$

where $\mathbb{E}^{\mathcal{S}_i}$ and $\mathbb{E}^{\mathcal{L}_j}$ denote the expectation over the data distribution of $(\mathbf{x}, y)$ in surrogate domain $\mathcal{S}_i$ and labeled domain $\mathcal{L}_j$ respectively and L_Y is the cross-entropy loss for multi-class classification. This term indicates that classifier h is shared by and trained on all surrogate domains.

The second term $\frac{1}{N}\sum_i \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{O}_i(e(X)), \mathcal{S}_i(e(X)))$ in Eqn. 1 leads to the second term of our objective function:

$$V_d(f, e, \boldsymbol{\alpha}) = \frac{1}{2N}\sum_{i=1}^{N} \{\mathbb{E}^{\mathcal{O}_i}[L_D(f(e(\mathbf{x}), i), 1)] + \sum_{j=1}^{N} \alpha_{i,j} \mathbb{E}^{\mathcal{L}_j}[L_D(f(e(\mathbf{x}), i), 0)]\}, \quad (3)$$

where $\mathbb{E}^{\mathcal{O}_i}$ denotes expectation over the data distribution of $(\mathbf{x}, y)$ in original domain $\mathcal{O}_i$. L_D is the cross-entropy loss for binary classification. f is a conditional feature discriminator that takes the feature $e(\mathbf{x})$ and the domain index i as input and classifies whether $e(\mathbf{x})$ comes from original domain $\mathcal{O}_i$ or surrogate domain $\mathcal{S}_i = \sum_j \alpha_{i,j} \mathcal{L}_j$. Note that $\min_f V_d(f, e, \boldsymbol{\alpha})$ measures how well the discriminator can distinguish between original domain $\mathcal{O}_i$ and surrogate domain $\mathcal{S}_i = \sum_j \alpha_{i,j} \mathcal{L}_j$, and therefore connects to the distance term $\frac{1}{N}\sum_i \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{O}_i(e(X)), \mathcal{S}_i(e(X)))$ in Eqn. 1.

The third term $\frac{1}{N}\sum_i \lambda_i$ in Eqn. 1 leads to the third term of our objective function:

$$V_\lambda(\{h_i\}_{i=1}^N, e, \boldsymbol{\alpha}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}^{\mathcal{S}_i}[L_Y(h_i \circ e(\mathbf{x}), y)] = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_{i,j} \mathbb{E}^{\mathcal{L}_j}[L_Y(h_i \circ e(\mathbf{x}), y)], \quad (4)$$

which is an upper bound of $\frac{1}{N}\sum_i \lambda_i$ in Eqn. 1 (more details in Sec. 1.2 in the Supplement). Different from the classifier h trained on all surrogate domains, h_i is the domain-specific classifier which is trained only on surrogate domain $\mathcal{S}_i$ (note that h_i is omitted in Fig. 2 for clarity).

Putting them all together, Eqn. 1 corresponds to the following minimax game as the final objective function:

$$\min_{h,e,\{h_i\}_{i=1}^N} \min_{\boldsymbol{\alpha}} \max_f V_h(h, e, \boldsymbol{\alpha}) - \lambda_d V_d(f, e, \boldsymbol{\alpha}) + V_\lambda(\{h_i\}_{i=1}^N, e, \boldsymbol{\alpha}), \quad (5)$$

where λ_d is a hyperparameter to balance loss terms (the performance of the classifier and feature alignment in particular), since too heavy penalization over the feature-level misalignment may harm the performance. In all experiments, λ_d is set to 1, which already achieves good performance.

Assigning Sub-Budget for Each Domain. After training converges in round r, we then assign budget such that domain j's proportion of the total budget until round r, $\beta_j^{(r)} \propto \sum_{i=1}^{N} \alpha_{i,j}$. In Sec. 4, we prove that this is the optimal sub-budget that minimizes the error bound.

3.5 Enhancing CAL with Augmented Instance-Level Acquisition

We propose a new instance-level strategy, dubbed **Gra**dient with **D**iscriminator **S**core (GraDS), which combines domain-level information with the gradient-based method, BADGE, to enhance our CAL. We choose to build GraDS on BADGE, since we found that combining CAL with BADGE is more effective than other strategies (more results in the Supplement). Our GraDS incorporates into BADGE the computed "outlier score" (i.e., our discriminator's output probability $f(Z,i) = \frac{\mathcal{O}_i(Z)}{\mathcal{O}_i(Z)+\mathcal{S}_i(Z)}$), which evaluates how much a data point looks like an outlier to the surrogate domain and therefore needs to be labeled in the next round. Multiplying the BADGE gradient with the outlier score $f(Z,i)$ leads to a revised gradient, which is then used with k-means++ [1] to select samples for labeling. GraDS prioritizes data points with higher outlier score ($f(Z,i)$) and classifier uncertainty (BADGE gradient) in each round to reduce classification error in the original domain. See more details on GraDS in Sec. 1.5 of the Supplement.

4 Theoretical Analysis

In this section, we provide theoretical analysis for CAL. We first provide an upper bound for the error of one original domain in Lemma 4.1, based on which we develop the upper bound for the average error over all original domains in Theorem 4.1. We then prove the optimality of CAL's sub-budget assignment strategy in Theorem 4.2. **All proofs are in Sec. 2 of the Supplement.**

Lemma 4.1 below provides theoretical guarantees that connect the prediction errors of $\mathcal{S}_i$ and $\mathcal{O}_i$.

Lemma 4.1 (**Error Bound for One Domain**). *Let $\mathcal{H}$ be a hypothesis space, and h, $h_i \in \mathcal{H} : \mathcal{Z} \to [0,1]$. $\mathcal{O}_i(Z)$ is the feature distribution of original domain i, and its surrogate domain $\mathcal{S}_i(Z) = \sum_j \alpha_{i,j}\mathcal{L}_j(Z)$ is a weighted average of N labeled domains $\{\mathcal{L}_j(Z)\}_{j=1}^N$. With the surrogate error $\epsilon_{\mathcal{S}_i}(h) = \sum_j \alpha_{i,j}\epsilon_{\mathcal{L}_j}(h)$ and $\lambda_i = \min_{h_i}(\epsilon_{\mathcal{O}_i}(h_i) + \epsilon_{\mathcal{S}_i}(h_i))$, we have:*

$$\begin{aligned}\epsilon_{\mathcal{O}_i}(h) &\le \epsilon_{\mathcal{S}_i}(h) + \tfrac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{O}_i(Z),\mathcal{S}_i(Z)) + \lambda_i \\ &= \sum_j \alpha_{i,j}\epsilon_{\mathcal{L}_j}(h) + \tfrac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{O}_i(Z),\sum_j \alpha_{i,j}\mathcal{L}_j(Z)) + \lambda_i\end{aligned}$$

Based on Lemma 4.1, Theorem 4.1 upper bounds the average error of all original domains.

Theorem 4.1 (**Error Bound for All Domains**). *Let $\mathcal{H}$ be a hypothesis space of VC dimension d and $h, h_i \in \mathcal{H} : \mathcal{Z} \to [0,1]$ be any hypothesis in $\mathcal{H}$. If labeled domains contain $\mathcal{M}$ data points in total, with $\beta_j\mathcal{M}$ assigned to labeled domain j, then for any $\delta \in (0,1)$, with probability at least $1-\delta$:*

$$\begin{aligned}\tfrac{1}{N}\textstyle\sum_i \epsilon_{\mathcal{O}_i}(h) \le \sum_j \alpha_j\epsilon_{\mathcal{L}_j}(h) + \tfrac{1}{2N}\sum_i d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{O}_i(Z),\mathcal{S}_i(Z)) \\ + \tfrac{1}{N}\textstyle\sum_i \lambda_i = \mathcal{U} \le \mathcal{U}_E\end{aligned} \tag{6}$$

where $\mathcal{U}_E$ is a further upper bound involving the empirical error $\sum_j \alpha_j\widehat{\epsilon}_{\mathcal{L}_j}(h)$:

$$\begin{aligned}\mathcal{U}_E = \sum_j \alpha_j\widehat{\epsilon}_{\mathcal{L}_j}(h) + 2\sqrt{(\sum_j \tfrac{\alpha_j^2}{\beta_j})(\tfrac{2d\log(2(\mathcal{M}+1)+\log(\frac{4}{\delta}))}{\mathcal{M}})} \\ + \tfrac{1}{2N}\sum_i d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{O}_i(Z),\mathcal{S}_i(Z)) + \tfrac{1}{N}\sum_i \lambda_i,\end{aligned} \tag{7}$$

where $\alpha_j = \frac{1}{N}\sum_i \alpha_{i,j}$, $\mathcal{S}_i(Z) = \sum_j \alpha_{i,j}\mathcal{L}_j(Z)$, $\sum_j \alpha_{i,j} = 1$, and $\lambda_i = \min_{h_i}(\epsilon_{\mathcal{O}_i}(h_i) + \epsilon_{\mathcal{S}_i}(h_i))$.

With the upper bound $\mathcal{U}$ in Eqn. 6 above, one can tighten the upper bound using $\min_{h,e}\min_{\boldsymbol{\alpha}} \mathcal{U}$, leading to CAL's objective function in Eqn. 5.

Note that in each round of multi-domain AL, domain j's accumulated budget ratio is β_j. Therefore one can search for the optimal β_j that minimizes the error bound in Eqn. 7. Theorem 4.2 below shows that the optimal sub-budget strategy is achieved when $\beta_j = \alpha_j = \frac{1}{N}\sum_i \alpha_{i,j}$.

Theorem 4.2 (**Optimal Budget Assignment**). *Assuming $\alpha_j > 0$ and $\beta_j > 0$ for $j = 1,2,...,N$, with $\sum_j \alpha_j = 1$ and $\sum_j \beta_j = 1$. The optimal upper bound for the average error of all domains, i.e., $\mathcal{U}_E$ in Eqn. 7, is achieved when $\beta_j = \alpha_j = \frac{1}{N}\sum_i \alpha_{i,j}$ for $j = 1,2,...,N$.*

Intuitively the optimal sub-budget assignment $\beta_j = \frac{1}{N}\sum_i \alpha_{i,j}$ implies that representative domains that are more similar (important) to other domains will receive a greater labeling budget.

5 Experiments

5.1 Baselines and Implementations

We compared our CAL with six AL baselines, including **Random** [31], **Margin** [32], **BADGE** [2], **Cluster-Margin** [8], **Energy** [44], and **BvSB-DA** [20]. Among them, **Random** is a simple strategy that randomly selects data points to label. **Energy** is a state-of-the-art hybrid query strategy for active domain adaptation (see Sec. 2 for differences between multi-domain AL and active domain adaptation); we adapt their query methods for our multi-domain AL setting. **BvSB-DA** combines domain adaptation (DA) and active learning. **Margin**, **BADGE**, and **Cluster-Margin** are state-of-the-art query strategies based on uncertainty and/or diversity (more details in the Supplement). Each baseline has two variants, **Joint**, which treats all data as a single domain and performs instance-level AL, and **Separate**, which assigns identical labeling budgets to each domain and performs instance-level AL in each domain.

Implementation. We run a model in $R = 5$ rounds plus an initial round, with three different random seeds, and report the average results over three seeds (see Sec. 1 of the Supplement for more details).

5.2 RotatingMNIST

To show insight and effectiveness of our methods, we begin with a synthetic dataset, RotatingMNIST-D6 with 6 domains, where original domain i contains images with rotation angles in the range $[(i-1)\times 30^\circ, i\times 30^\circ)$. The training

Table 1: RotatingMNIST results (%). "Joint" and "Separate" indicate joint and separate assignment, respectively (see details in Sec. 5.1). We mark the best results with **bold face**.

Query	Random		Margin		BADGE		Cluster-Margin		Energy		BvSB-DA		CAL (Ours)
	Joint	Separate	Joint	Separate	Joint	Separate	Joint	Separate	Joint	Separate	Joint	Separate	
Round 0	49.3	49.2	49.3	49.2	49.3	49.2	49.3	49.2	49.3	49.2	47.1	47.1	**51.7**
Round 1	59.3	58.4	59.9	59.3	59.6	61.0	59.7	58.7	59.8	59.8	58.3	57.6	**71.8**
Round 2	65.2	64.5	65.4	65.8	65.9	66.2	66.0	65.6	65.5	64.7	62.8	63.2	**81.0**
Round 3	69.4	68.8	69.7	70.6	69.7	71.4	70.1	70.0	69.7	69.2	68.0	69.0	**85.7**
Round 4	77.7	77.7	80.5	80.3	79.2	80.5	80.2	80.6	79.2	80.1	74.1	74.2	**87.8**
Round 5	79.9	80.5	81.9	83.1	82.5	83.1	82.2	83.3	82.1	82.9	77.4	76.9	**87.9**
Average	66.8	66.5	67.8	68.1	67.7	68.6	67.9	67.9	67.6	67.6	64.6	64.6	**77.7**

Table 2: Office-Home results (%). "Joint" and "Separate" indicate joint and separate assignment, respectively (see details in Sec. 5.1). We mark the best results with **bold face**.

Query	Random		Margin		BADGE		Cluster-Margin		Energy		BvSB-DA		CAL (Ours)
	Joint	Separate	Joint	Separate	Joint	Separate	Joint	Separate	Joint	Separate	Joint	Separate	
Round 0	32.8	32.8	32.8	32.8	32.8	32.8	32.8	32.8	32.8	32.8	31.0	31.0	**34.5**
Round 1	42.0	42.8	41.3	41.6	42.1	42.6	41.4	42.8	41.5	41.2	37.6	38.0	**46.3**
Round 2	48.3	48.6	47.4	46.6	48.3	48.1	46.4	47.0	47.7	48.0	44.6	44.8	**52.8**
Round 3	52.4	52.6	50.5	51.3	51.9	52.0	50.0	50.6	51.4	51.2	48.3	48.3	**57.4**
Round 4	56.9	57.6	55.2	55.9	57.3	57.1	54.9	55.9	56.2	56.0	51.4	53.2	**59.6**
Round 5	57.7	58.0	55.4	57.7	57.7	56.9	56.0	57.5	58.1	58.4	52.9	53.9	**60.9**
Average	48.4	48.7	47.1	47.6	48.3	48.2	46.9	47.8	48.0	47.9	44.3	44.9	**51.9**

and test sets of each domain contain 10000 and 1666 images, respectively. The total labeling budget per round m is 150, which is 0.25% of the training set.

Accuracy. Table 1 shows the accuracy for different methods. One interesting observation is that the performance of joint assignment and separate assignment is very similar for each baseline method. Additionally, Table 1 reveals that directly aligning the original domains as BvSB-DA does can actually harm performance; in the DA setting, aligning source and target domains improves performance because source domains have plenty of labeled data and therefore provide valuable classification-relevant information for target domains; this is *not* the case for our AL setting because most data points are unlabeled; therefore, insufficient labeled data of the original domains in BvSB-DA lead to inaccurate estimation of domain classification accuracy, which introduces misleading classification-relevant information to each domain, leading to a decrease in overall classification accuracy. It is also worth noting that our full method CAL achieves the highest accuracy, surpassing BvSB-DA and the five instance-level methods (Random, Margin, BADGE, Cluster-Margin, and Energy) by up to 13.1% and 11.2%, respectively, in terms of average accuracy across all rounds.

Estimated Domain Similarity $\alpha_{i,j}$. To gain more insights on CAL's domain selection step, we visualize the estimated similarity weights $\alpha_{i,j}$ in Fig. 3. Specifically, Fig. 3 shows CAL's estimated similarity matrix $\boldsymbol{\alpha} = [\alpha_{i,j}]_{i=1,j=1}^{6,6}$ on RotatingMNIST-D6 and the corresponding sub-budget for each domain $\beta_j = \alpha_j = \frac{1}{6}\sum_i \alpha_{i,j}$. We observe that (1) nearby domains have larger $\alpha_{i,j}$, which makes sense since they have similar rotation angles, (2) more budget is assigned to the middle domains, which is expected since labeling middle domains improves performance for domains on both sides, (3) $\alpha_{1,6}$ and $\alpha_{6,1}$ are larger, which is because

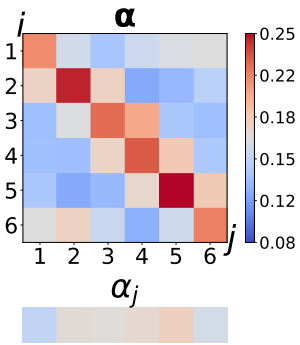

Figure 3: Estimated similarity $\alpha_{i,j}$ and α_j for RotatingMNIST-D6.

digits 0, 1, and 8 are identical after they are rotated by around 180° (see Sec. 3 in the Supplement for more results on visualizing $\alpha_{i,j}$, including visualization on more datasets and the evolution of the similarity matrix from round 0 to round 5).

5.3 Real-World Datasets

We use three real-world datasets: Office-Home (65 classes) [41], ImageCLEF (12 classes) [30], and Office-Caltech (10 classes) [12]. Each dataset consists of four domains (for more details, refer to Section 1.4 of the Supplement). We split each dataset into training and test sets. In each round, we allocate a labeling budget of 200, 20, and 20, respectively for the three datasets. This budget represents approximately 1% of the training set for each dataset. Tables 2∼4 present the accuracy (%) for different methods. Notably, our CAL demonstrates significant improvements compared to the baselines. Similar to the findings for RotatingMNIST, it is expected that BvSB-DA performs worse than the instance-level baselines. In contrast, our CAL consistently

Table 3: ImageCLEF results (%). "Joint" and "Separate" indicate joint and separate assignment, respectively (see details in Sec. 5.1). We mark the best results with **bold face**.

Query	Random		Margin		BADGE		Cluster-Margin		Energy		BvSB-DA		CAL (Ours)
	Joint	Separate	Joint	Separate	Joint	Separate	Joint	Separate	Joint	Separate	Joint	Separate	
Round 0	39.4	39.4	39.4	39.4	39.4	39.4	39.4	39.4	39.4	39.4	35.8	35.8	**41.7**
Round 1	52.9	49.8	58.2	58.8	54.4	54.0	56.1	52.2	54.2	58.0	53.0	49.7	**61.2**
Round 2	60.0	57.8	64.6	61.7	62.2	58.5	59.5	59.9	63.2	63.3	58.4	57.0	**69.5**
Round 3	66.2	65.0	68.8	69.0	66.4	64.7	65.8	65.6	66.5	70.2	63.3	66.5	**75.3**
Round 4	70.0	69.8	74.3	71.2	73.1	68.0	71.9	70.0	71.5	72.7	69.3	70.7	**74.9**
Round 5	69.4	69.5	72.0	71.5	71.9	70.3	71.0	71.0	71.8	71.1	66.6	67.5	**77.1**
Average	59.7	58.6	62.9	62.0	61.2	59.2	60.6	59.7	61.1	62.5	57.7	57.9	**66.6**

Table 4: Office-Caltech results (%). "Joint" and "Separate" indicate joint and separate assignment, respectively (see details in Sec. 5.1). We mark the best results with **bold face**.

Query	Random		Margin		BADGE		Cluster-Margin		Energy		BvSB-DA		CAL (Ours)
	Joint	Separate	Joint	Separate	Joint	Separate	Joint	Separate	Joint	Separate	Joint	Separate	
Round 0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	47.0	47.0	**53.0**
Round 1	73.7	77.6	72.1	72.5	74.5	68.8	70.9	69.0	70.5	74.5	65.0	67.3	**80.7**
Round 2	79.4	80.9	79.0	81.7	80.3	78.9	77.5	81.1	82.5	82.9	79.7	76.2	**89.3**
Round 3	85.4	85.3	84.6	86.3	86.8	87.7	83.7	87.8	88.1	87.9	83.9	86.5	**91.6**
Round 4	89.3	88.4	89.6	91.5	89.0	91.7	88.6	90.4	90.4	90.3	87.8	90.0	**93.2**
Round 5	90.6	90.4	90.3	91.0	89.5	92.4	91.3	89.3	87.7	91.0	88.4	89.2	**93.0**
Average	78.1	78.8	77.6	78.8	78.3	78.3	77.0	77.9	78.2	79.4	75.3	76.1	**83.5**

Table 5: Ablation study (%) for CAL. 2^{nd} and 3^{rd} denote the second and the third terms of Eqn. 5, respectively. CAL+BADGE denotes the combination of domain-level CAL and BADGE. We mark the best results with **bold face**.

Method	CAL (Ours)	CAL+ BADGE	w/o 3^{rd}	w/o 2^{nd}	w/o 2^{nd} & 3^{rd}	BADGE
Round 0	**41.7**	41.7	43.4	36.7	39.6	39.4
Round 1	**61.2**	58.3	55.5	58.5	55.3	54.0
Round 2	**69.5**	67.8	63.5	63.3	59.7	58.5
Round 3	**75.3**	71.2	70.7	68.5	69.8	64.7
Round 4	**74.9**	71.2	72.1	71.0	71.2	68.0
Round 5	**77.1**	74.9	76.7	73.3	71.2	70.3
Average	**66.6**	64.2	63.7	61.9	61.1	59.2

outperforms the instance-level baselines (Random, Margin, BADGE, Cluster-Margin, and Energy) across all five AL rounds.

5.4 Ablation Studies

CAL's Components. To investigate the impact of different components in CAL, we conduct an ablation study using the ImageCLEF dataset. Table 5 shows the results. The full CAL's accuracy drops by 2.4% if our instance-level stragegy GraDS is replaced with BADGE, demonstrating the effectiveness of our GraDS. Moreover, removing the third and second terms leads to performance drops of 2.9% and 4.7% respectively. This highlights the effectiveness of the second term in reducing distribution shift across domains and the benefits of upper bounding λ_i in the third term, rather than ignoring it. Furthermore, removing both the third and second terms leads to a larger performance drop of 5.1%. The results in Table 5 indicate that all CAL's components contribute to its performance improvement over the state of the art.

Adapting Active Domain Adaptation Methods for MUDAL. In Sec. 2, we provide a thorough comparison between multi-domain active learning (MUDAL) and domain adaptation (DA). We conduct experiments using several state-of-the-art (active) DA methods to highlight the distinctions between MUDAL and (active) DA. Consistent with the findings for BvSB-DA, Table 4 in the Supplement reveals that directly aligning the original domains can actually lead to a decline in performance. For more detailed information, please refer to Section 3 of the Supplement.

Please refer to Sec. 3 in the Supplement for more visualization results, baselines, and ablation studies.

6 Conclusion and Future Work

We identify the problem of multi-domain active learning, propose the first general AL method that integrates domain-level and instance-level information, and provide both detailed theoretical analysis and empirical results. Our work demonstrates the effectiveness of our proposed CAL (and its simplified variants) on multi-domain data and shows its potential for significant real-world applications.

Furthermore, our CAL method has the potential for future applications in Natural Language Processing (NLP). There are studies that have utilized active learning in *single-domain* NLP to improve classification and in-context learning [9, 29]. With our CAL, these methodologies can potentially be adapted to multi-domain settings with promising performance; this is thanks to our CAL's compatibility with instance-level active learning methods.

Acknowledgements: Wang and Gao would like to acknowledge funding support (CCF-2118953, CCF-2208663, DMS-2311064, DMS-2220271, IIS-2127918).

References

[1] Arthur, D.; and Vassilvitskii, S. 2007. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 1027–1035.

[2] Ash, J. T.; Zhang, C.; Krishnamurthy, A.; Langford, J.; and Agarwal, A. 2020. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*.

[3] Bachman, P.; Sordoni, A.; and Trischler, A. 2017. Learning algorithms for active learning. In *international conference on machine learning*, 301–310. PMLR.

[4] Balcan, M.-F.; Beygelzimer, A.; and Langford, J. 2009. Agnostic active learning. *Journal of Computer and System Sciences*, 75(1): 78–89.

[5] Baram, Y.; Yaniv, R. E.; and Luz, K. 2004. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5(Mar): 255–291.

[6] Ben-David, S.; Blitzer, J.; Crammer, K.; Kulesza, A.; Pereira, F.; and Vaughan, J. W. 2010. A theory of learning from different domains. *Machine learning*, 79(1): 151–175.

[7] Berlind, C.; and Urner, R. 2015. Active nearest neighbors in changing environments. In *International conference on machine learning*, 1870–1879. PMLR.

[8] Citovsky, G.; DeSalvo, G.; Gentile, C.; Karydas, L.; Rajagopalan, A.; Rostamizadeh, A.; and Kumar, S. 2021. Batch active learning at scale. *Advances in Neural Information Processing Systems*, 34.

[9] Dor, L. E.; Halfon, A.; Gera, A.; Shnarch, E.; Dankin, L.; Choshen, L.; Danilevsky, M.; Aharonov, R.; Katz, Y.; and Slonim, N. 2020. Active learning for BERT: an empirical study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 7949–7962.

[10] Du, X.; Zhong, D.; and Shao, H. 2019. Building an active palmprint recognition system. In *2019 IEEE International Conference on Image Processing (ICIP)*, 1685–1689. IEEE.

[11] Ducoffe, M.; and Precioso, F. 2018. Adversarial active learning for deep networks: a margin based approach. *arXiv preprint arXiv:1802.09841*.

[12] Fernando, B.; Habrard, A.; Sebban, M.; and Tuytelaars, T. 2014. Subspace alignment for domain adaptation. *arXiv preprint arXiv:1409.5241*.

[13] Fu, B.; Cao, Z.; Wang, J.; and Long, M. 2021. Transferable query selection for active domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7272–7281.

[14] Fu, M.; Yuan, T.; Wan, F.; Xu, S.; and Ye, Q. 2021. Agreement-discrepancy-selection: active learning with progressive distribution alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 7466–7473.

[15] Gal, Y.; Islam, R.; and Ghahramani, Z. 2017. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, 1183–1192. PMLR.

[16] Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; Marchand, M.; and Lempitsky, V. 2016. Domain-adversarial training of neural networks. *JMLR*, 17(1): 2096–2030.

[17] Geifman, Y.; and El-Yaniv, R. 2017. Deep active learning over the long tail. *arXiv e-prints*, arXiv–1711.

[18] Gissin, D.; and Shalev-Shwartz, S. 2019. Discriminative active learning. *arXiv preprint arXiv:1907.06347*.

[19] Hanneke, S.; et al. 2014. Theory of disagreement-based active learning. *Foundations and Trends® in Machine Learning*, 7(2-3): 131–309.

[20] He, R.; Liu, S.; He, S.; and Tang, K. 2022. Multi-domain active learning: literature review and comparative study. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(3): 791–804.

[21] Hsu, W.-N.; and Lin, H.-T. 2015. Active learning by learning. In *Twenty-Ninth AAAI conference on artificial intelligence*.

[22] Huang, S.-J.; Jin, R.; and Zhou, Z.-H. 2010. Active learning by querying informative and representative examples. *Advances in neural information processing systems*, 23.

[23] Joshi, A. J.; Porikli, F.; and Papanikolopoulos, N. 2009. Multi-class active learning for image classification. In *2009 ieee conference on computer vision and pattern recognition*, 2372–2379. IEEE.

[24] Kremer, J.; Steenstrup Pedersen, K.; and Igel, C. 2014. Active learning with support vector machines. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(4): 313–326.

[25] Li, L.; Jin, X.; Pan, S. J.; and Sun, J.-T. 2012. Multi-domain active learning for text classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1086–1094.

[26] Liu, T.; Xu, Z.; He, H.; Hao, G.; Lee, G.-H.; and Wang, H. 2023. Taxonomy-Structured Domain Adaptation. In *ICML*.

[27] Liu, Z.; Ding, H.; Zhong, H.; Li, W.; Dai, J.; and He, C. 2021. Influence selection for active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9274–9283.

[28] Ma, X.; Gao, J.; and Xu, C. 2021. Active universal domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 8968–8977.

[29] Margatina, K.; Schick, T.; Aletras, N.; and Dwivedi-Yu, J. 2023. Active learning principles for in-context learning with large language models. *arXiv preprint arXiv:2305.14264*.

[30] National Bureau of Statistics. 2014. ImageCLEF dataset. https://www.imageclef.org/2014/adaptation/.

[31] Ren, P.; Xiao, Y.; Chang, X.; Huang, P.-Y.; Li, Z.; Gupta, B. B.; Chen, X.; and Wang, X. 2021. A survey of deep active learning. *ACM Computing Surveys (CSUR)*, 54(9): 1–40.

[32] Roth, D.; and Small, K. 2006. Margin-based active learning for structured output spaces. In *European Conference on Machine Learning*, 413–424. Springer.

[33] Saha, A.; Rai, P.; Daumé, H.; Venkatasubramanian, S.; and DuVall, S. L. 2011. Active supervised domain adaptation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 97–112. Springer.

[34] Schohn, G.; and Cohn, D. 2000. Less is more: Active learning with support vector machines. In *ICML*, volume 2, 6.

[35] Sener, O.; and Savarese, S. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.

[36] Settles, B. 2009. Active learning literature survey. TR1648, University of Wisconsin-Madison Department of Computer Sciences.

[37] Shi, H.; and Wang, H. 2023. A Unified Approach to Domain Incremental Learning with Memory: Theory and Algorithm. In *NeurIPS*.

[38] Sinha, S.; Ebrahimi, S.; and Darrell, T. 2019. Variational adversarial active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5972–5981.

[39] Su, J.-C.; Tsai, Y.-H.; Sohn, K.; Liu, B.; Maji, S.; and Chandraker, M. 2020. Active adversarial domain adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 739–748.

[40] Tur, G.; Hakkani-Tür, D.; and Schapire, R. E. 2005. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45(2): 171–186.

[41] Venkateswara, H.; Eusebio, J.; Chakraborty, S.; and Panchanathan, S. 2017. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5018–5027.

[42] Wang, D.; and Shang, Y. 2014. A new active labeling method for deep learning. In *2014 International joint conference on neural networks (IJCNN)*, 112–119. IEEE.

[43] Wang, H.; He, H.; and Katabi, D. 2020. Continuously Indexed Domain Adaptation. In *International Conference on Machine Learning*, 9898–9907. PMLR.

[44] Xie, B.; Yuan, L.; Li, S.; Liu, C. H.; Cheng, X.; and Wang, G. 2021. Active learning for domain adaptation: An energy-based approach. *arXiv preprint arXiv:2112.01406*.

[45] Xu, Z.; Hao, G.; He, H.; and Wang, H. 2023. Domain Indexing Variational Bayes: Interpretable Domain Index for Domain Adaptation. In *ICLR*.

[46] Xu, Z.; Lee, G.-H.; Wang, Y.; Wang, H.; et al. 2022. Graph-Relational Domain Adaptation. In *ICLR*.

[47] Zhang, Z.; Jin, X.; Li, L.; Ding, G.; and Yang, Q. 2016. Multi-domain active learning for recommendation. In *Thirtieth AAAI Conference on Artificial Intelligence*.