



The Hebrew University of Jerusalem
The Rachel and Selim Benin School of Computer Science and Engineering

Active Learning for All Learners

Inbal Mishal

Thesis submitted in partial fulfillment of the requirements
for the Master of Sciences degree
in Computer Science

Under the supervision of **Prof. Daphna Weinshall**

June 2024



האוניברסיטה העברית בירושלים
בית הספר להנדסה ולמדעי המחשב על שם רחל וסלים בנין
החוג למדעי המחשב

למידה אקטיבית המותאמת לכל המודלים הלומדים

מוגש על ידי
ענבל משעל

עבודת גמר לתואר מוסמך במדעי המחשב

עבודה זו הונחתה על ידי
פרופ' דפנה ויינשל

יוני 2024

תקציר

טכניקות למידה אקטיבית בעולם של למידה עמוקה הראו שיפורים משמעותיים בהורדת עלויות התיוג עבור אימון מודלים. למרות זאת, היעילות שלהם במצבים עם תקציבים משתנים נותרה אתגר ולא קיים אלגוריתם יחיד שמספק פתרון עבור איזורי תקציב שונים כשהמושג "תקציב" מתייחס לכמות הדגימות המתוייגות הנתונות למודל הלומד. מחקר זה מציג שיטת למידה אקטיבית חדשנית שמגשרת על הפער הזה.

בניגוד לשיטות הקיימות, השיטה המוצגת כאן משנה באופן דינאמי את המדד לפיו היא בוחרת דגימות ומתבססת על הביצועים והיכולות של המודל הנוכחי. גישה זו מתבססת על מחקר תיאורטי שמעיד על הצורך בשילוב פשטות וייצוגיות עם חוסר ודאות. באמצעות ניסויים רבים על מאגרי מידע מגוונים, ואתגור השיטה עם שיטות למידה שונות, אנו מדגימים את היכולת של האלגוריתם שלנו להתגבר על בעיית ההתחלה הקרה ולהגיע לביצועים הטובים ביותר.

Abstract

Deep Active Learning (AL) techniques can be effective in reducing annotation costs for training deep models. However, their effectiveness in low- and high-budget scenarios seems to require different strategies, and achieving optimal results across varying budget scenarios remains a challenge. In this study, we introduce Dynamic Coverage Margin mix (*DCoM*), a novel active learning approach designed to bridge this gap.

Unlike existing strategies, *DCoM* dynamically adjusts its strategy, considering the competence of the current model. Through theoretical analysis and empirical evaluations on diverse datasets, including challenging computer vision tasks, we demonstrate *DCoM*'s ability to overcome the cold start problem and consistently improve results across different budgetary constraints. Thus *DCoM* achieves state-of-the-art performance in both low- and high-budget regimes.

Acknowledgements

I am deeply grateful to my supervisor, Prof. Daphna Weishall, for her guidance and support throughout this thesis.

Additionally, I am deeply grateful to Dr. Guy Hacoen, whose expertise greatly enhanced the quality and depth of this research.

I also thank the Hebrew University for its resources.

To Daphna's lab, thank you for your valuable input.

I would like to thank my family, friends, and colleagues for their encouragement, assistance, and moral support.

To all those mentioned above and to anyone else who has contributed in any way, however small, I offer my heartfelt thanks.

Contents

1	Introduction	1
1.1	Background and Motivation	1
1.2	Active Learning Types	2
2	Related Work	7
2.1	Uncertainty-Based Active Learning	7
2.2	Diversity-Based Active Learning	8
2.3	Uncertainty and Diversity Combination	8
2.4	The Cold Start Problem	9
2.5	Division into Low and High Budget Strategies	9
2.6	Self-Supervised Methods for Representation Learning	10
3	Theoretical Analysis	12
3.1	Preliminaries	14
3.2	Refined error bound	15
3.3	Competence score $S_{\mathcal{L}}$	16

4	Our Method - <i>DCoM</i>	18
4.1	Active learning method	18
4.2	Time and space complexity of <i>DCoM</i>	23
4.2.1	Selection of initial δ	23
4.2.2	Customizing δ values	24
4.2.3	Active sampling	25
5	Empirical Results	26
5.1	Methodology	26
5.2	Implementation details	27
5.2.1	Supervised training	29
5.2.2	Semi-supervised training	29
5.2.3	Self-supervised feature extraction	30
5.3	Main Results	31
5.4	Additional empirical results	34
5.4.1	δ_0 Initialization	34
5.4.2	Δ distribution throughout the training process	35
5.4.3	Dynamic algorithm coverage	35
5.4.4	Decomposing the algorithm	36
5.4.5	Different embedding space	38
5.4.6	Different uncertainty-based methods.	38
5.4.7	Comparison per dataset	41
5.5	Hyper-parameters exploring	44

<i>CONTENTS</i>	vi
6 Constrained K-Means Variation	45
6.1 Background	45
6.2 COPAL-Kmeans: Kmeans Variation For Active Learning	46
6.3 Combining COPAL-Kmeans in <i>DCoM</i>	47
6.3.1 Empirical Results	49
6.3.2 Discussions	51
7 Summary And Conclusions	52

List of Figures

1.1	Diagram illustrating the three main active learning scenarios, [36]	2
1.2	Coverage comparison of each algorithm using the SimCLR representation on the STL-10 dataset, visualized via t-SNE projection with 250 examples. You can see that <i>DCoM</i> achieves more precise coverage and encompasses a larger number of points compared to Prob Cover.	6
5.1	Mean accuracy difference between several AL algorithms and random queries. Positive mean difference indicates that the corresponding AL method is beneficial. The errors bars correspond to standard error, based on 5-10 repetitions (3 for the ImageNet subsets). While some methods perform well only in specific budget regimes, <i>DCoM</i> consistently achieves the best results across all budgets. Cumulative budget is evenly spaced for easy range comparison.	31

- 5.2 Comparison of AL strategies in a semi-supervised learning framework. Each bar represents the mean test accuracy over 3 repetitions of FlexMatch, trained using a total of 400 (left) and 1,000 (right) labeled examples on the CIFAR-100 dataset (an average of 4 and 10 labeled examples per class respectively). Three rounds of AL are considered, selecting 100 examples in the first round, 300 in the second, and 600 in the third. 33
- 5.3 $\pi(\delta)$, estimated from the unlabeled data and using k-means algorithm for labeling. The dashed line indicates the highest δ , after which purity drops below $\alpha = 0.95$ 34
- 5.4 The Δ distribution through *DCoM* algorithm. Each plot displays the mean and standard error over 5-10 repetitions. The results indicate that as the emphasis shifts towards maximizing the purity of sample balls, rather than solely focusing on maximum coverage, the radii adapt accordingly. This, combined with the improved performance observed with *DCoM*, highlights the necessity of the dynamic algorithm in enhancing the effectiveness of active sampling. 35
- 5.5 Comparison between $P(C(\mathbb{L}, \Delta))$ values during *DCoM* sampling process, across several datasets. Each plot displays the mean and standard error over 3 – 10 repetitions. These figures demonstrates a consistent behavior of probability coverage. 36

5.6 Performance evaluation (mean accuracy and STE) when optimizing each component of objective function (3.1) separately and together. DPC denotes the Dynamic Probability Coverage, corresponding to $\mathcal{O}_{low}(x)$ in *DCoM*. Margin is *DCoM*'s choice for $\mathcal{O}_{high}(x)$. Each plot displays the mean and standard error over 5 – 10 repetitions. 37

5.7 Assessment of each algorithm's objective function individually across multiple datasets. DPC stands for Dynamic Probability Coverage, corresponding to $\mathcal{O}_{low}(x)$ in *DCoM*. *DCEM* represents *DCoM* with entropy as $\mathcal{O}_{high}(x)$ instead of margin. Each plot displays the mean and standard error over 5 – 10 repetitions. These graphs demonstrate how the weighted objective function consistently produces superior results across all labeled set sizes $|\mathbb{L}|$. . . 37

5.8 Performance over CIFAR-100, using different embedding spaces for *DCoM* and *ProbCover*, see details in the caption of Fig. 5.1. Clearly, *DCoM* consistently achieves the best results. 38

5.10 Similar to Fig. 5.1 and Tables 5.2-5.3, using only *DCoM* while leveraging 4 different variants for its underlying uncertainty score. Each variation incorporates a dynamic algorithm weighted with a different uncertainty-based method. Performance is assessed by the accuracy difference between each variation and no active learning (random sampling). Each plot corresponds to 3 – 5 repetitions using the CIFAR-100 dataset. These findings suggest that when the algorithm begins with optimization on $\mathcal{O}_{low}(x)$, the performance differences between the variants are small. 40

- 5.11 Evaluation of *DCoM* across different settings of the logistic function hyper-parameters a and k . Each plot displays the mean and standard error over 3 – 5 repetitions. The findings indicate that fine-tuning these parameters has negligible effects on performance. 44

- 6.1 Mean accuracy difference between several AL algorithms and random queries. Positive mean difference indicates that the corresponding AL method is beneficial. The errors bars correspond to standard error, based on 5-10 repetitions (3 for the ImageNet subsets). Here you can see that the COPAL-Kmeans variation didn't improve the performance. 50

List of Tables

5.1	Initial delta values used in experiments	28
5.2	Model final accuracy when varying the size of the labeled set \mathbb{L} (row) and the respective AL strategy (column), using the ImageNet-100 dataset	32
5.3	Same as Table 5.2, using the CIFAR-100 dataset.	33
5.4	Model accuracy for different sizes of labeled set \mathbb{L} and several uncertainty-based methods for $\mathbb{F}(x)$	40
5.5	Model accuracy for different sizes of labeled set \mathbb{L} and AL strategies	42
5.6	Model accuracy for different sizes of labeled set \mathbb{L} and AL strategies	43

1 Introduction

1.1 Background and Motivation

Deep Learning (DL) algorithms require a lot of data to achieve optimal results. In certain scenarios, there is an abundance of unlabeled data but limited capacity for labeling it. In fields such as medical imaging, an invaluable resource — doctors themselves — serves as a costly oracle. Pool-based Active learning algorithms aim to address this challenge by reducing the burden of labeling and making it more effective. Unlike traditional supervised learning frameworks, where the model passively learns from a given dataset, Active Learning (AL) can affect the construction of the labeled set, possibly by leveraging knowledge about the current learner. Accordingly, the initial goal of AL is to select q examples to be annotated, where q represents the number of examples that can be sent to the oracle. It aims to leverage the current knowledge of the learner to choose the examples that will most effectively enhance the learner's performance. AL has already demonstrated tangible contributions across various domains such as computer vision tasks [48], NLP [26, 38] and medical imaging [15]. These examples highlight the importance of advancing AL to achieve even greater impact.

Over the past years, active learning has been an active area of research [12, 36,

39, 42]. The choice of an AL strategy depends on both the learner’s inductive biases and the nature of the problem. Recent research suggests that the optimal active learning strategy varies with the budget size, where budget refers to the size of the training set. When the budget is large, methods based on uncertainty and diversity sampling are most effective. Conversely, when the budget is small, methods centered on typicality and diversity are more appropriate. However, there is no single active learning strategy that is suitable for all budget regimes. Our study aims to address this challenge by dynamic selection of the best examples based on the current state of the learner and the current budget.

1.2 Active Learning Types

The realm of Active Learning (AL) is segmented into three problem settings: membership query synthesis, stream-based selective sampling, and pool-based active learning [36].

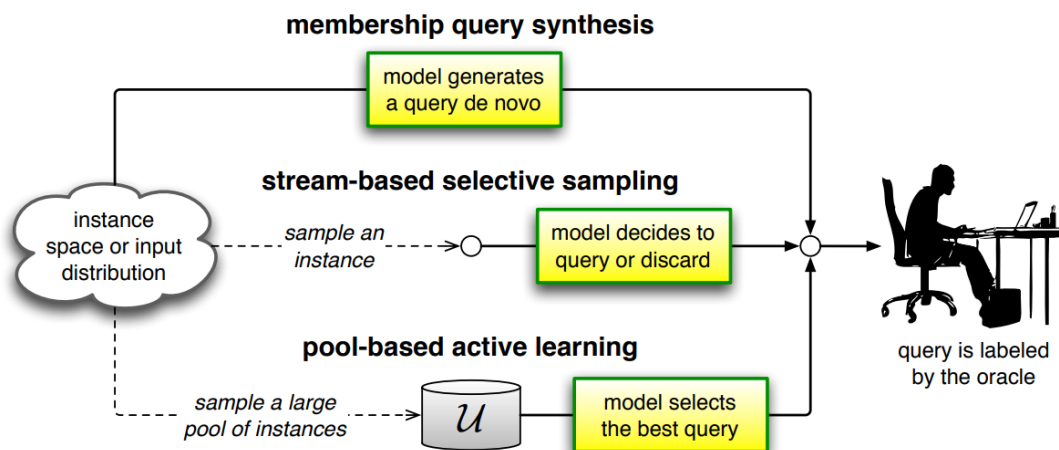


Figure 1.1: Diagram illustrating the three main active learning scenarios, [36]

Membership Query Synthesis: One of the earliest active learning scenarios to be explored is learning with membership queries [1]. In this scenario, the learner can request labels for any unlabeled instance in the input space, including queries it generates from scratch rather than those sampled from a natural distribution. Efficient query synthesis is often feasible and effective for finite problem domains [2].

While query synthesis can be practical for many problems, it becomes challenging when the oracle is a human annotator. For instance, Baum and Lang [5] used membership query learning with human oracles to train a neural network for classifying handwritten characters. They encountered an issue where many generated query images contained no recognizable symbols, only artificial hybrid characters lacking natural semantic meaning. Similarly, membership queries in natural language processing might generate nonsensical text or speech. To address these limitations, stream-based and pool-based scenarios (discussed in the next sections) have been proposed.

Despite these challenges, [27] describe an innovative and promising application of the membership query scenario in a real-world setting. They developed a “robot scientist” capable of conducting autonomous biological experiments to discover metabolic pathways in the yeast *Saccharomyces cerevisiae*. Here, an instance consists of a mixture of chemical solutions forming a growth medium and a specific yeast mutant, with the label indicating whether the mutant thrived. All experiments are autonomously synthesized using an active learning approach based on inductive logic programming and performed by a laboratory robot. This method reduced the cost of experimental materials three-fold compared to naively running the least expensive experiment and by a factor of 100 compared to randomly generated experiments. In domains where labels are derived from experiments rather than human annotators, query syn-

thesis holds promise for automated scientific discovery.

Stream-Based Selective Sampling: An alternative to synthesizing queries is selective sampling [11]. This method assumes that obtaining an unlabeled instance is either free or inexpensive, allowing instances to be sampled from the actual distribution first, and then the learner decides whether to request their labels. This approach is often referred to as stream-based or sequential active learning, where each unlabeled instance is drawn one at a time from the data source, and the learner must choose to query or discard it. If the input distribution is uniform, selective sampling may behave similarly to membership query learning. However, if the distribution is nonuniform and unknown, queries will still be sensible since they originate from a real underlying distribution. The decision to query an instance can be framed in several ways, as described in [36].

Pool-Based Active Learning: Pool-based active learning involves a large pool of unlabeled data \mathbb{U} and a smaller set of initially labeled data \mathbb{L} . In this setting, AL algorithms operate within a given budget (b) and their goal is to select the most informative instances from \mathbb{U} for labeling, aiming to improve the learning model effectively. This process may be repeated iteratively, gradually increasing budget b . This selection typically involves assessing each instance's potential informativeness using measures such as uncertainty sampling, query-by-committee, or expected model change. These measures help rank the instances in \mathbb{U} by their potential to enhance the model's performance if labeled. The top-ranked instances are then selected for annotation, and their labels are acquired from an oracle, such as human annotators or other reliable sources. By iteratively adding newly labeled instances to \mathbb{L} , pool-based active learning aims to optimize learning efficiency and model accuracy, particularly beneficial in scenarios with abundant unlabeled data and constrained labeling resources. The main difference between stream-based and pool-based active learning is

that the former processes data sequentially, making query decisions individually, whereas the latter evaluates and ranks the entire collection before selecting the best query. While pool-based scenarios are more common in application papers, stream-based approaches may be more suitable in contexts with limited memory or processing power, such as mobile and embedded devices. The majority of current AL research adheres to the pool-based AL setting, which we specifically investigate in our study.

As briefly reviewed below, recent research often categorizes active sampling methods according to the budget regime. In contrast, our study takes a different approach by recognizing the need to adapt sample selection methods based on the learner's competence. We aim to develop a single algorithm that dynamically adjusts its sampling strategy in response to the learner's evolving abilities. This transition from budget to learner's competence represents a significant advancement in active learning research, as it allows us to move beyond rigid budget allocations and towards a more adaptive approach that optimizes performance based on the learner's abilities.

In this thesis, we introduce a novel method called Dynamic Coverage & Margin mix (*DCoM*), which dynamically adjusts its selection strategy to deliver optimal results. *DCoM* utilizes the coverage of the labeled set as a metric of progress and leverages the learning model to enhance the accuracy of this coverage measure. Fig.1.2 compares the coverage of Prob Cover with *DCoM* coverage and illustrates how our algorithm significantly improves coverage accuracy.

We begin by providing a simplified theoretical framework (Chapter 3), where we integrate a typicality test with an uncertainty test based on the budget and its coverage. Inspired by the insights gained from this analysis, we introduce *DCoM* (Chapter 4), which combines these characteristics. *DCoM* aims to offer

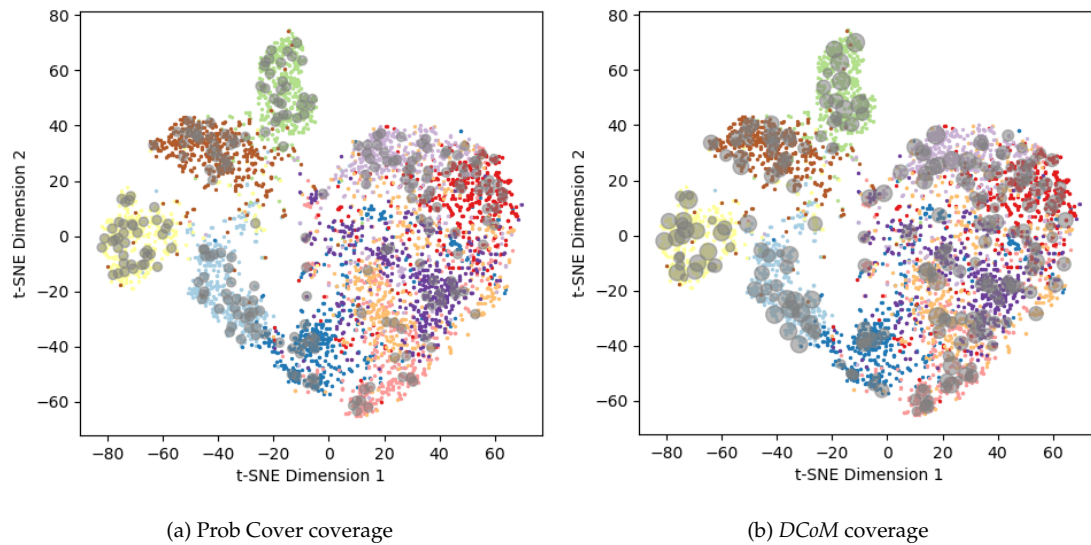


Figure 1.2: Coverage comparison of each algorithm using the SimCLR representation on the STL-10 dataset, visualized via t-SNE projection with 250 examples. You can see that *D*CoM achieves more precise coverage and encompasses a larger number of points compared to Prob Cover.

a flexible solution for any budget by selecting the most appropriate examples that are most suitable to the current budget. We validate *D*CoM through an extensive empirical study using various vision datasets (Chapter 5). We provide an initial investigation into the impact of constrained K-means on *D*CoM (Chapter 6). Our findings indicate that *D*CoM effectively achieves superior performance across all budget ranges.

2 Related Work

2.1 Uncertainty-Based Active Learning

In some Active Learning (AL) methods, the selection of the most informative data points relies on uncertainty measure. The underlying principle is that the most informative examples are those about which the classifier is least certain.

Uncertainty sampling was first introduced by Lewis and Gale [30] for efficient text classifier training. Subsequent methods employed various known measures, such as entropy [37], least confidence (max softmax output), and minimal margin (lowest margin between the two highest softmax outputs) to implement AL sampling based on uncertainty measures.

Over time, new AL algorithms were developed exclusively on the uncertainty sampling principle. In the field of Bayesian active learning, the selection process is guided by Bayesian principles, utilizing probability distributions to manage uncertainty [16, 23, 45, 51].

A notable approach by Ranganathan et al. [34] introduces a loss function that leverages both labeled and unlabeled data, aiming to minimize cross-entropy for labeled data while reducing entropy for unlabeled data. Another method by Gissin and Shalev-Shwartz [17] selects examples to label in a manner that

makes the labeled set and the unlabeled pool indistinguishable. Additionally, Cho et al. [8] propose the least disagree metric (LDM), defined as the smallest probability of disagreement of the predicted label, as a new uncertainty metric.

2.2 Diversity-Based Active Learning

In these AL methods, the algorithm prioritizes diverse sampling over uncertainty. Some algorithms use an embedding space to represent data points. Hu et al. [24] employs clustering to select the densest point initially, and then iteratively chooses the next densest point that is farthest from the previously selected point. Sener and Savarese [35] define the problem of active learning as core-set selection, which involves selecting a subset of points such that a model trained on this subset performs competitively on the remaining data points. Hachohen et al. [19] define the concept of typicality and use clustering to select the most typical and diverse samples. Yehuda et al. [47] adopt the coverage perspective of Sener and Savarese [35], but instead of aiming to cover the entire space, they use a given ball radius and clustering to select the most covering points.

2.3 Uncertainty and Diversity Combination

Several algorithms combine both uncertainty and diversity in their approaches. For instance, Kirsch et al. [28] and Ash et al. [4] integrate these two aspects in their methodologies. Yang et al. [46] selects the most uncertain samples while incorporating a diversity constraint in the objective function. Wen et al. [44] employs kernel methods to estimate the empirical risk of each sample using k-means labeling and uses the Neural Tangent Kernel to approximate the classifier trained on self-supervised features. Chen et al. [6] frame the task as two

problems: biased sampling between different classes and selecting atypical examples within the classes. They address the first by using k-means to sample an equal number of points from each cluster and tackle the second by using augmentations to compute a new confidence score. Chowdhury et al. [9] utilize supervised contrastive learning, training the representation model during the AL process, and employ k-means for selecting examples.

2.4 The Cold Start Problem

Over time, the cold start problem emerged, highlighting the challenges faced by AL algorithms when operating with a small budget. This problem arises because traditional active learning algorithms, which primarily rely on uncertainty methods, are less efficient than random selection during the initial stages. It seems that *uncertainty* and *diversity* do not fully capture this challenge. Chen et al. [6] describe the cold start problem as a combination of two issues: unbalanced sampling across different classes (related to diversity) and unbalanced sampling within each class, where uncertain examples are initially prioritized over certain ones.

2.5 Division into Low and High Budget Strategies

Some studies categorize the world of AL problems into two regimes of budget strategies - low and high, where 'budget' refers to the amount of available labeled data. They argue that to solve the cold start problem, different strategies should be used depending on the current budget regime. In low-budget scenarios, typical examples often yield the best outcomes for the model [6, 19, 20, 47]. This implies that in such contexts, AL algorithms should aim to maximize both

certainty and diversity. Conversely, in high-budget scenarios, the focus shifts towards maximizing uncertainty while maintaining diversity. [20] proposes a method to select the most suitable approach based on the current state. This method involves comparing multiple active learning strategies during runtime, rather than recommending a single universal approach.

2.6 Self-Supervised Methods for Representation Learning

Self-Supervised Learning (SSL) methods for representation learning aim to extract meaningful features from data without requiring manually labeled examples. These techniques leverage inherent data structures or relationships to generate supervised-like signals for training. There are several SSL methods used for image representation learning. In this research, we primarily utilized SimCLR [7]. Additionally, we employed MOCOv2 [21], BYOL [18], Barlow Twins [49], and DINOv2 [33].

SimCLR (Simple Contrastive Learning): SimCLR is a powerful self-supervised learning framework designed to learn effective data representations. It operates by maximizing agreement between differently augmented views of the same data instance while minimizing agreement between views of different instances. This approach encourages the model to learn invariant features that capture semantic information useful for downstream tasks. SimCLR has shown promising results across various domains, demonstrating its ability to learn robust representations that generalize well.

SimCLR exemplifies how self-supervised methods can effectively learn representations by leveraging contrastive learning principles. These methods col-

lectively advance the field of representation learning by enabling models to capture meaningful and transferable features from unlabeled data. We employ these methods to ensure that distances reflect semantic dissimilarities accurately.

3 Theoretical Analysis

The framework of active learning adopted here can be formalized as follows: Let \mathbb{X} denote the input domain, \mathbb{Y} the target domain, and $\mathcal{L} : \mathbb{X} \rightarrow \mathbb{Y}$ the target learner. Before seeing any labels, \mathcal{L} may be a random feasible hypothesis, or it may be initialized by either transfer learning or self training.

Active learning may involve a single step, or an iterative process with repeated active learning steps. In each step of active learning, the learner actively seeks labels by choosing a set of unlabeled points as a query set, to be labeled by an oracle/teacher. Subsequently, this set of labeled points is added to the learner's supervised training set, and the learner is retrained or fine-tuned.

As discussed in the introduction, it has been shown that different active learning strategies are suitable for different budgets b . Henceforth, b refers to the number of labeled examples known to the learner at the beginning of an active learning step. With a low budget b , strategies that do not rely on the outcome of the learner are most suitable. When b is high, it is beneficial to take into consideration the confidence of the learner when choosing an effective query set. What makes a budget high or low was left vague in previous analysis, as it clearly depends on the specific application and dataset.

To achieve an effective active learning protocol suitable for all learners, irrespective of budget, we aim to devise a universal objective function, whose min-

imization is used to select the query set. Next, we discuss its envisioned form for selecting the query set at the beginning of each AL step.

We begin our discussion by proposing to replace the notion of *budget* with the notion of *competence*: a network is competent if its generalization score is relatively high, and vice versa. This proposal is more formally justified in Section 3.3. We then rephrase the intuition stated above as follows: When the learner’s competence is low, the objective function should prioritize typicality and diversity of selected queries (see discussion in the introduction) irrespective of the learner’s predictions. When the learner’s competence is high, its uncertainty in prediction should be given high priority.

More formally, let $\mathcal{O}(x)$ denote the desired objective function for query selection, where $x \in \mathbb{X}$ is unlabeled. Let $\mathcal{O}_{low}(x)$ and $\mathcal{O}_{high}(x)$ denote the objective functions suitable for a learner with low competence and high competence respectively. Let $S_{\mathcal{L}}$ denote a score, which captures the competence of learner \mathcal{L} . The proposed objective function can now be written as follows $\forall x \in \mathbb{X}$:

$$\mathcal{O}(x) = (1 - S_{\mathcal{L}}) \cdot \mathcal{O}_{low}(x) + S_{\mathcal{L}} \cdot \mathcal{O}_{high}(x) \quad (3.1)$$

In Section 3.2 we discuss the design of $\mathcal{O}_{low}(x)$. In order to tackle the challenge of active learning when the competence of \mathcal{L} is low, we utilize the point coverage framework introduced in [47]. This approach relies on self-supervised data representation, which is blind to the learner’s performance. We refine their analysis and obtain an improved generalization bound for the 1-Nearest Neighbor (1-NN) classification model that depends on the local geometry of the data. Beforehand, in Section 3.1 we discuss the necessary preliminaries required to introduce the coverage approach. In Section 3.3 we discuss our choice of $S_{\mathcal{L}}$, and connect it to the notion of budget discussed in previous work.

We select $\mathcal{O}_{high}(x)$ to equal one minus the normalized lowest response between

the two highest softmax outputs of \mathcal{L} at x , or *Margin*, as this is a common measure of prediction uncertainty.

3.1 Preliminaries

Notations. Let P denote the underlying probability distribution of data \mathbb{X} . Assume that a true labeling function $f : \mathbb{X} \rightarrow \mathbb{Y}$ exists. Let $\mathbb{U} \subseteq \mathbb{X}$ denote the unlabeled set of points, and $\mathbb{L} \subseteq \mathbb{X}$ the labeled set, such that $\mathbb{X} = \mathbb{U} \cup \mathbb{L}$. Here $|\mathbb{L}| = b \leq m$ is the annotation budget where $|\mathbb{X}| = m$.

Let $B_\delta(x) = \{x' \in \mathbb{X} : \|x' - x\|_2 \leq \delta\}$ denote a ball of radius δ centered at x . Let $C \equiv C(\mathbb{L}, \delta) = \bigcup_{x \in \mathbb{L}} B_\delta(x)$ denote the region covered by δ -balls centered at the labeled examples in \mathbb{L} . We call $C(\mathbb{L}, \delta)$ the *covered region*, where $P(C)$ denotes its probability. Let f^N denote the 1-NN classifier, and \mathcal{L} denote our current learner – a 1-NN classifier based on \mathbb{L} .

Definition. We say that a ball $B_\delta(x)$ is *pure* if $\forall x' \in B_\delta(x) : f(x') = f(x)$.

Definition. We define the *purity* of δ as

$$\pi(\delta) = P(\{x \in \mathbb{X} : B_\delta(x) \text{ is pure}\}).$$

Notice that $\pi(\delta)$ is monotonically decreasing, as can be readily verified.

In [47] it is shown that the generalization error of the 1-NN classifier f^N is bounded as follows

$$\mathbb{E} \left[f^N(x) \neq f(x) \right] \leq (1 - P(C(\mathbb{L}, \delta))) + (1 - \pi(\delta)). \quad (3.2)$$

Subsequently, an algorithm is proposed that minimizes the first term in (3.2) by maximizing the coverage probability $P(C(\mathbb{L}, \delta))$, while ignoring the second term that is assumed to be fixed.

Below, we begin by refining the bound, focusing more closely on its second term $(1 - \pi(\delta))$. This is used in Chapter 4 to devise an improved algorithm that minimizes simultaneously both terms of the refined error bound.

3.2 Refined error bound

Define the indicator random variable $I_\delta(x) = \mathbb{1}_{\{x \in \mathbb{X} : B_\delta(x) \text{ is pure}\}}$. By definition,

$$\pi(\delta) = P(\{x \in \mathbb{X} : B_\delta(x) \text{ is pure}\}) = \mathbb{E}[I_\delta(x)].$$

Since the distribution of \mathbb{X} is not known a priori, we use the *empirical distribution* to approximate the expected value of this random variable. In accordance, given a labeled set $\mathbb{L} = \{x_i\}_{i=1}^b$:

$$\begin{aligned} \hat{\pi}(\delta) &= \hat{\mathbb{E}}[\mathbb{1}_{\{x \in \mathbb{L} : B_\delta(x) \text{ is pure}\}}] = \frac{1}{b} \mathbb{E}\left[\sum_{i=1}^b \mathbb{1}_{\{B_\delta(x_i) \text{ is pure}\}}\right] = \frac{1}{b} \sum_{i=1}^b \mathbb{E}[\mathbb{1}_{\{B_\delta(x_i) \text{ is pure}\}}] \\ &= \frac{1}{b} \sum_{i=1}^b P(B_\delta(x_i) \text{ is pure}). \end{aligned} \tag{3.3}$$

With this approximation

$$\mathbb{E}\left[f^N(x) \neq f(x)\right] \leq \left[1 - P(C(\mathbb{L}, \delta))\right] + \left[1 - \frac{1}{b} \sum_{i=1}^b P(B_\delta(x_i) \text{ is pure})\right] + \varepsilon. \tag{3.4}$$

where ε bounds the error of the empirical approximation in (3.3).

Note that the refined bound in (3.4) depends on the purity separately at each labeled point in \mathbb{L} . We further refine this bound by allowing the fixed radius δ to be chosen individually, where δ_i denotes the radius of point $x_i \in \mathbb{L}$. Let $\Delta = [\delta_i]_{i=1}^b$ denote the list of individual radii corresponding to $\mathbb{L} = \{x_i\}_{i=1}^b$. The cover defined by Δ is $C(\mathbb{L}, \Delta) = \bigcup_{(x_i, \delta_i) \in \mathbb{L} \times \Delta} B_{\delta_i}(x_i)$.

We consider the *normalized Nearest Neighbor* (nNN) classification algorithm, where the nearest neighbor is chosen by solving $f^N(x) = \operatorname{argmin}_{x_i \in \mathbb{L}} \frac{d(x, x_i)}{\delta_i}$. The bound on the error in (3.4) becomes

$$\mathbb{E} \left[f^N(x) \neq f(x) \right] \leq \left[1 - P(C(\mathbb{L}, \Delta)) \right] + \left[1 - \frac{1}{b} \sum_{i=1}^b P(B_{\delta_i}(x_i) \text{ is pure}) \right] + \varepsilon. \quad (3.5)$$

3.3 Competence score $S_{\mathcal{L}}$

In order to obtain a useful competence score¹, which can be effectively used in objective function (3.1), we require that it satisfies the following conditions:

- (i) Lie in the range $[0, 1]$.
- (ii) Use all the data.
- (iii) Monotonically increase with the competence of learner \mathcal{L} .

In accordance, we propose to use the probability of coverage:

$$S_{\mathcal{L}}(\mathbb{L}, \Delta) = \frac{1 + e^{-k(1-a)}}{1 + e^{-k(P(C(\mathbb{L}, \Delta)) - a)}} \quad (3.6)$$

This function follows a logistic growth curve, with parameters defined as follows: $a \in (0, 1)$ is the midpoint and $k \in (0, \infty)$ determines the curve's steepness.

First, we note that this score satisfies the conditions stated above for the nNN classifier: (i)-(ii), as it is a function in $[0, 1]$ that depends on all the data \mathbb{X} , while (iii) follows from the error bound (3.5). Below, we prove that it is also monotonically increasing with budget b .

¹When the labeled set \mathbb{L} is large, it may be possible to set aside a validation set to directly estimate the competence of learner \mathcal{L} , but this is not feasible when the labeled set \mathbb{L} is small.

More specifically, prior work showed empirically and theoretically that the training budget $b = |\mathbb{L}|$ provides a useful clue for the desirable trade-off between $\mathcal{O}_{low}(x)$ and $\mathcal{O}_{high}(x)$ in (3.1), which is captured by score $S_{\mathcal{L}}(\mathbb{L}, \Delta)$ defined in (3.6). This score is monotonically increasing with $P(C(\mathbb{L}, \Delta))$ according to the logistic function definition and the parameters constraints. We next show that $S_{\mathcal{L}}(\mathbb{L}, \Delta)$ is monotonically increasing with b . This implies that definition (3.6) agrees with empirical evidence in prior work.

proposition: For two labeled sets \mathbb{L}, \mathbb{L}' , if $\mathbb{L} \subseteq \mathbb{L}'$ then $S_{\mathcal{L}}(\mathbb{L}, \Delta) \leq S_{\mathcal{L}}(\mathbb{L}', \Delta')$.

Proof. First, we note that

$$\mathbb{L} \subseteq \mathbb{L}' \implies C(\mathbb{L}, \Delta) = \bigcup_{x_i \in \mathbb{L}} B_{\delta_i}(x_i) \subseteq \left(\bigcup_{x_i \in \mathbb{L}} B_{\delta_i}(x_i) \right) \cup \left(\bigcup_{x_i \in \mathbb{L}' \setminus \mathbb{L}} B_{\delta_i}(x_i) \right) = C(\mathbb{L}', \Delta')$$

Since probability is monotonic, it follows that $P(C(\mathbb{L}, \Delta)) \leq P(C(\mathbb{L}', \Delta'))$. From (3.6), and by the definition of the logistic function using the provided parameters limits, $S_{\mathcal{L}}(\mathbb{L}, \Delta)$ is monotonically increasing in \mathbb{L} , which implies that $S_{\mathcal{L}}(\mathbb{L}, \Delta) \leq S_{\mathcal{L}}(\mathbb{L}', \Delta')$. \square

4 Our Method - *DCoM*

The error bound in (3.5) guides our method for active query selection, which aims to minimize this bound by maximizing the coverage $P(C(\mathbb{L}, \Delta))$ and individual purity $\pi(\delta_i) = P(B_{\delta_i}(x_i) \text{ is pure})$. Recall that \mathbb{L} denotes the set of labeled points, δ_i the influence radius of $x_i \in \mathbb{L}$, $\Delta = [\delta_i]_{i=1}^b$, and

$$C(\mathbb{L}, \Delta) = \bigcup_{(x_i, \delta_i) \in \mathbb{L} \times \Delta} B_{\delta_i}(x_i), \quad \pi(\delta_i) = P(\{x \in B_{\delta_i}(x_i) : f(x) = f(x_i)\})$$

Since it was shown in [47] that maximizing $P(C(\mathbb{L}, \delta))$ is already NP-hard, we opt for developing a greedy algorithm for actively selecting the query set, to be termed *DCoM*.

4.1 Active learning method

The algorithm involves several steps:

Initialization: Define a suitable embedding space for the data and initialize essential parameters.

If $\mathbb{L} \neq \emptyset$, train model \mathbb{M} on \mathbb{L} , and compute δ -**expansion** on the elements of \mathbb{L} . Otherwise, $\Delta = [\delta_i = \delta_0]_{i=1}^b$.

Repeat /* iterative active learning steps */

- (i) **Active query selection:** determine which examples to select for annotation.
- (ii) **Model training:** obtain labels and train a deep learning model \mathbb{M} .
- (iii) **δ -expansion:** use \mathbb{M} to update δ_i for the recently selected examples.

Until the accumulated budget is exhausted.

We next provide a detailed description of each step.

Initialization: We begin by seeking a suitable embedding space using $\mathbb{U} \cup \mathbb{L}$, where distances are intended to be inversely correlated with semantic similarity. Self-supervised or representation learning may be used to this end. We choose an initial radius δ_0 as described in [47].

(i) **Active query selection:** In this phase (see Alg 1 below for pseudo-code), the algorithm selects q examples from \mathbb{U} to send for labeling, where q denotes the query set size. To this end, it first computes several parameters:

Algorithm 1 *DCoM, Active sampling*

Input: unlabeled pool \mathbb{U} , labeled pool \mathbb{L} , query size q , list of ball sizes Δ , trained model \mathbb{M} if $\mathbb{L} \neq \emptyset$

Output: a set of points to query, and the coverage of \mathbb{L} ,

$\forall x \in \mathbb{U}, M(x) \leftarrow$ normalized margin between the two highest softmax outputs of \mathbb{M} if $\mathbb{L} \neq \emptyset$, 1 otherwise

Compute $P(C(\mathbb{L}, \Delta))$

Compute $S_{\mathcal{L}}(\mathbb{L}, \Delta)$

$\delta_{\text{avg}} \leftarrow \text{Average}(\Delta)$

$G = (V = \mathbb{U} \cup \mathbb{L}, E = \{(x, x') : x' \in B_{\delta_{\text{avg}}}(x)\})$

for $(x_i, \delta_i) \in \mathbb{L} \times \Delta$ **do**

$\forall x \in B_{\delta_i}(x_i): \forall e = (v, x) \in E$, remove e # DON'T COVER POINTS TWICE

$\forall e = (x_i, v) \in E$, remove e

end for

$Q \leftarrow \emptyset$

for $i = 1, \dots, q$ **do**

$\forall x \in \mathbb{U}, \text{ODR}(x) \leftarrow$ Out-Degree Rank

$\forall x \in \mathbb{U}, R(x) \leftarrow S_{\mathcal{L}}(\mathbb{L}, \Delta) \cdot (1 - M(x)) + (1 - S_{\mathcal{L}}(\mathbb{L}, \Delta)) \cdot \text{ODR}(x)$

$x_{\text{max}} \leftarrow \{\text{argmax}_{x \in \mathbb{U}} R(x)\}$

$Q \leftarrow Q \cup \{x_{\text{max}}\}$

$\forall x \in B_{\delta_{\text{avg}}}(x_{\text{max}}): \forall e = (v, x) \in E$, remove e

$M(x_{\text{max}}) \leftarrow 1$

end for

return $Q, P(C(\mathbb{L}, \Delta))$

- $M(x)$, $\forall x \in \mathbb{U}$: the normalized margin between the two highest softmax outputs using the last trained model if $\mathbb{L} \neq \emptyset$, 1 otherwise.
- Δ : $\Delta = [\delta_i = \delta_0]_{i=1}^b$ if $\mathbb{L} = \emptyset$, otherwise use Δ from the last AL iteration.
- δ_{avg} : the average of Δ .

- $P(C(\mathbb{L}, \Delta))$: the probability of coverage of the current \mathbb{L} with its corresponding Δ .
- $S_{\mathcal{L}}(\mathbb{L}, \Delta)$: the competence score of objective function (3.6) if $\mathbb{L} \neq \emptyset$, 0 otherwise.
- $\{G_{\delta_{\text{avg}}} = (V, E)\}$: a directed adjacency graph where nodes represent samples and edges connect pairs of nodes if the distance between them in the embedding space is smaller than δ_{avg} . In order not to cover points more than once, $\forall x_i \in \mathbb{L}$, $\forall x \in B_{\delta_i}(x_i)$ and $\forall v \in V$ – prune incoming edges $e = (v, x) \in E$. Additionally, we prune all outgoing edges $e = (x_i, v) \in E$.

Next, q points are to be selected from \mathbb{U} in a greedy manner as follows:

1. $\forall x \in \mathbb{U}$, compute the Out-Degree-Rank $\text{ODR}(x)$.
2. $\forall x \in \mathbb{U}$, compute a ranking score $R(x) = S_{\mathcal{L}}(\mathbb{L}, \Delta) \cdot (1 - M(x)) + (1 - S_{\mathcal{L}}(\mathbb{L}, \Delta)) \cdot \text{ODR}(x)$.
3. Select the vertex x_{max} with the highest score, $x_{\text{max}} = \text{argmax}_{x \in \mathbb{U}} R(x)$.
4. Remove all incoming edges to x_{max} and its neighbors, implying that $\text{ODR}(x_{\text{max}}) = 0$.
5. Set $M(x_{\text{max}}) = 1$.

(ii) Model training:

1. Obtain labels for query set Q , returned in step (i) of **Active sampling**.
2. Remove set Q from \mathbb{U} and add it to \mathbb{L} .
3. Train the given model \mathbb{M} using \mathbb{L} (and \mathbb{U} if the learner is semi-supervised).

(iii) δ -expansion for new samples. In this step (see Alg 2 below for pseudocode), we compute a suitable δ_i for each new labeled example $x_i \in Q$, and add it to Δ . First, we use the updated model M to predict labels for all the

points in the unlabeled set \mathbb{U} . We also compute a purity threshold $\tau = m \cdot P(C(\mathbb{L} \setminus Q, \Delta)) + b$, where m and b are hyperparameters and $P(C(\mathbb{L} \setminus Q, \Delta))$ - the probability of coverage using the old labeled set - has been computed in the first **Active sampling** step. For each new labeled example $v \in Q$, we search for the largest radius δ_{opt} such that its purity $P\left(\left\{x \in B_{\delta_{\text{opt}}}(v) : f(x) = f(v)\right\}\right)$ is still larger than τ . Assuming that $\pi(B_{\delta}(x))$ a monotonic with δ , we can leverage binary search.

Algorithm 2 *DCoM, δ -expansion*

Input: unlabeled pool \mathbb{U} , labeled pool \mathbb{L} , query pool Q , maximal ball size δ_{max} , list of ball sizes Δ for $\mathbb{L} \setminus Q$, trained model \mathbb{M} from the current iteration and $P(C(\mathbb{L} \setminus Q, \Delta))$ from **Active sampling**
Output: updated list of ball sizes Δ

```

 $\tau \leftarrow m \cdot P(C(\mathbb{L} \setminus Q, \Delta)) + b$ 
 $\hat{\mathbb{Y}} \leftarrow$  The predicted label for each  $x \in \mathbb{U}$  using the model  $\mathbb{M}$ 
for  $v \in Q$  do
     $\delta_{\text{opt}} \leftarrow \text{argmax}_{\delta} [P(\{x \in B_{\delta}(v) : f(x) = f(v)\}) > \tau]$ 
     $\Delta \leftarrow \Delta + [\delta_{\text{opt}}]$ 
end for
return  $\Delta$ 

```

The algorithm relies on $P(C(\mathbb{L}, \Delta))$ as an indicator of progress, which adjusts the purity threshold and affect the transition from a density-based scoring method to an margin-based one. Using a sparse representation of the adjacency graph enables *DCoM* to handle large datasets efficiently without exhaustive space requirements. The algorithm's complexity, including adjacency graph construction and sample selection, is discussed in Section 4.2.

4.2 Time and space complexity of *DCoM*

During neural network training, *DCoM* is invoked once to determine the optimal subset for human annotation, which is then utilized for network training.

Prior to active sampling, a preliminary step involves selecting the initial δ for the first adjacency graph. This procedure is detailed in [47].

Following this initial step and preceding the active sampling, the algorithm proceeds with customizing appropriate δ values for each labeled example from the previous iteration.

After this step, the active sampling commences and encompasses several subsequent stages. The comprehensive process is elaborated upon in Chapter 4.

For the complexity analysis below, let n represent the number of examples in the combined unlabeled and labeled pool $|\mathbb{L} \cup \mathbb{U}|$, d the dimension of the data embedding space, b - the given budget ($|\mathbb{L}|$) and q - the size of the query set. As previously stated, *DCoM* can be divided into three distinct steps:

4.2.1 Selection of initial δ

Time Complexity: The process begins by creating t adjacency graphs, each corresponding to a different δ value being evaluated. Here, t denotes the number of δ 's being examined. The time complexity of each graph generation is $O(n^2 \cdot d)$ time. The computation of purity requires the prediction of pseudo-labels by k-means at a cost of $O(n^2)$. The complexity of computing ball purity for each example is $O(n^2)$, so totally each the ball purity computing takes $O(n^3)$. This adds up to overall time complexity of $O(tdn^3)$. Although it may seem significant, this step doesn't occur during the active learning algorithm process. Addi-

tionally, it's possible to confine the data employed for purity computation to the densest points, effectively lowering the complexity to $O(tdn^2)$. This approach mandates the use of at least the number of points equivalent to the number of classes for purity computation. This technique yields similar δ_0 values (consistent for CIFAR-10, CIFAR-100, and ImageNet-50, and with a variation of 0.05 for ImageNet-100/200 and STL-10 and 0.15 for SVHN).

Space complexity: Naively, the space complexity is $O(n^2)$, which might be impractical for large datasets like ImageNet. However, [47] demonstrates that utilizing a sparse matrix in coordinate list (COO) format and using limited δ values, the space complexity becomes $O(|E|)$, where E represents the set of edges in the graph. Although $O(|E|)$ remains $O(n^2)$ in the worst-case scenario, in practice, the average vertex degree with a limited radius is less than n .

4.2.2 Customizing δ values

Time complexity: Each iteration of *DCoM* starts with customizing δ values for each labeled example from the last active step. In the worst case, there are $|\mathbb{L}|$ examples. Customizing δ involves building the adjacency graph using δ_{\max} , running all examples on the model to obtain their predictions, and applying binary search over δ values between 0 and δ_{\max} for each example from the last active step. The binary search is performed on a continuous parameter (δ) with a search resolution r . This implies that there are $\frac{\delta_{\max}}{r}$ available values for δ . Overall, this process takes $O(dn^2 + n + |\mathbb{L}| \cdot \log(\frac{\delta_{\max}}{r})) = O(dn^2 + b \cdot \log(\frac{\delta_{\max}}{r}))$. In practical terms, due to the vectorization of these processes, it requires approximately 27 minutes to construct an adjacency graph for ImageNet-200 and update deltas for 1000 examples on a single CPU.

Space complexity: Similar to the space complexity analysis in 4.2.1, which is

$O(n^2)$.

4.2.3 Active sampling

Time complexity: Active sampling involves balancing model min-margin and example density based on the current adjacency graph. Initially, the margin of each example is computed using the softmax output from the model’s last layer, requiring $O(n)$ time. Subsequently, the current adjacency is created using δ_{avg} , which is the average over the \mathbb{L} δ values list - Δ . As discussed in 4.2.1, this step has a time complexity of $O(dn^2)$. Following this, the iterative process for selecting a single sample includes the following steps:

- Calculating node degrees – $O(|E|)$ time.
- Finding the node with maximal degree – $O(n)$ time.
- Removing incoming edges from the graph for covered points – $O(|E|)$ time.

Samples are iteratively selected from the current sparse graph, with incoming edges to newly covered samples being removed. Unlike adjacency graph creation, sample selection cannot be parallelized, as each step depends on the previous one. Overall, the time complexity for selecting a sample is $O(|E| + n)$, resulting in a worst-case overall complexity of $O(dn^2)$. However, as more points are selected, the removal of edges speeds up the selection of later samples. Practically, it consumes about 15 minutes to construct an adjacency graph and select 1000 samples from ImageNet-200 on a single CPU.

Space complexity: Similar to the previous space complexity analysis – $O(n^2)$.

5 Empirical Results

5.1 Methodology

This section investigates two deep AL frameworks: a *fully supervised* approach, wherein a deep model is trained exclusively on the annotated dataset \mathbb{L} as a standard supervised task, and a *semi-supervised* framework trained on both the annotated and unlabeled datasets, \mathbb{L} and \mathbb{U} . Our experimental setup is based on the codebase of [31], ensuring a fair comparison among the various AL strategies by using the same network architectures and sharing all relevant experimental conditions.

More specifically, in the following experiments, we trained ResNet-18 [22] on STL-10 [10], SVHN [32], CIFAR-100 [29] and subsets of ImageNet [14] including ImageNet-50, ImageNet-100, and ImageNet-200 as they appear in [40]. The hyper-parameters we used are detailed in Section 5.2. In the semi-supervised framework, we used FlexMatch [50] and the code provided by [43]. For the experiments, we followed the parameters specified by [50]. The exact parameters are detailed in Section 5.2. While the ResNet-18 architecture may no longer achieve state-of-the-art results on the datasets examined, it serves as a suitable platform to evaluate the efficacy of AL strategies in a competitive and fair environment, where they have demonstrated benefits.

In our comparisons, we used several active learning strategies (or optimization criteria) for selecting a query set: (1) *Random* sampling, (2) *Min margin* — lowest margin between the two highest softmax outputs, (3) *Max entropy* — highest entropy of softmax outputs [37], (4) *Uncertainty* — 1 minus maximum softmax output, (5) *DBAL* [16], (6) *CoreSet* [35], (7) *BALD* [28], (8) *BADGE* [4], (9) *ProbCover* [47], and (10) *LDM-s* [8]. When available, we used the code provided in [31] for each strategy. When it was unavailable, we used the code from the repositories of the corresponding papers. Since [8] doesn't provide code, we compared our results with the corresponding available results reported in their paper, despite differences in the running setup (see 5.4.7).

DCoM and *ProbCover* require a suitable data embedding. For STL-10, SVHN, CIFAR-10 and CIFAR-100 we employed the SimCLR [7] embedding. For the ImageNet subsets we used the DINOv2 [33] embedding. Below, we provide an extensive ablation study, which includes alternative embedding spaces. Moreover, it includes comparison between multiple uncertainty-based methods, revealing that the high-budget algorithm doesn't exert significant influence when initialized with the dynamic probability coverage method. Additionally, we demonstrate the efficacy of blending different methods by decomposing the algorithm into its two objective functions. In all experiments, identical settings and hyper-parameters are used for our method, as detailed in Section 5.2. The robustness of our method to these choices is discussed in Section 5.5.

5.2 Implementation details

The source code used in this study will be accessible at [47] GitHub repository. Our code will be made public upon acceptance.

DCoM necessitates the computation of the initial delta for each dataset representation. The values utilized in our experiments can be found in Table 5.1. The δ values of CIFAR-10 and CIFAR-100 are sourced from [47]. Information on the computation of δ values for other datasets can be located in 5.4.1.

Table 5.1: Initial delta values used in experiments

Dataset	SSL method	Initial δ
STL-10	SimCLR	0.55
SVHN	SimCLR	0.4
CIFAR-10	SimCLR	0.75
CIFAR-100	SimCLR	0.65
ImageNet-50	DINOv2	0.75
ImageNet-100	DINOv2	0.7
ImageNet-200	DINOv2	0.65
CIFAR-100	MOCOv2+	0.5
CIFAR-100	BYOL	0.5
CIFAR-100	Barlow Twins	0.55

Additionally, throughout our experiments we used identical algorithm parameters, distinguishing between those for datasets with smaller class amounts (10) and larger ones (50+). These parameters encompass the adaptive purity threshold τ from Alg 2 and the logistic function parameters for the competence score $S_{\mathcal{L}}(\mathbb{L}, \Delta)$ in (3.6), (a, k) . Specifically, we employed $\tau = 0.2$ cover + 0.4, with logistic parameters $a = 0.9$ and $k = 30$ for CIFAR-10, SVHN, and STL-10, and $a = 0.8$ and $k = 30$ for CIFAR-100 and ImageNet subsets. The δ resolution for the binary search in *DCoM* was set to 0.05. Our ablation study, detailed in Section 5.5, demonstrates that these selections have negligible impact on per-

formance.

5.2.1 Supervised training

When training on STL-10, SVHN, CIFAR-10 and CIFAR-100 datasets, we utilized a ResNet-18 architecture trained for 200 epochs. Our optimization strategy involved using an SGD optimizer with a Nesterov momentum of 0.9, weight decay set to 0.0003, and cosine learning rate scheduling starting at a base rate of 0.025. Training was performed with a batch size of 100 examples, and we applied random cropping and horizontal flips for data augmentation. For an illustration of these parameters in use, refer to [31].

When training ImageNet-50, we used the same hyper-parameters, only changing the base learning rate to 0.01, the batch size to 50 and the epoch amount to 50.

When training ImageNet-100/200, we used the same hyper-parameters as ImageNet-50, only changing the epoch amount to 100.

The train and test partitions followed the original sets from the corresponding paper, with 10% of the train set used for validation in all datasets except for ImageNet-200, where 5% was used for validation.

5.2.2 Semi-supervised training

When training FlexMatch [50], we used the semi supervised framework by [43], and the AL framework by [31]. All experiments involved 3 repetitions.

We relied on the standard hyper-parameters used by FlexMatch [50]. Specifically, we trained WideResNet-28 for 512 epochs using the SGD optimizer, with 0.03 learning rate, 64 batch size, 0.9 SGD momentum, 0.999 EMA momentum,

0.001 weight decay and 2 widen factor.

5.2.3 Self-supervised feature extraction

STL-10, SVHN, CIFAR-10, CIFAR-100. To extract semantically meaningful features, we trained SimCLR using the code provided by [40] for STL-10, SVHN, CIFAR-10 and CIFAR-100. Specifically, we used ResNet-18 [22] with an MLP projection layer to a 128-dim vector, trained for 512 epochs. All the training hyper-parameters were identical to those used by SCAN (all details can be found in [40]). After training, we used the 512 dimensional penultimate layer as the representation space.

Representation learning: ImageNet-50/100/200. We extracted features from the official (ViT-S/14) DINOv2 weights pre-trained on ImageNet [33]. We employed the L2-normalized penultimate layer for the embedding, which has a dimensionality of 384.

Ablation embeddings – CIFAR-100. We extracted more features as BYOL [18], MOCOv2+ [21], and Barlow Twins [49] using pre-trained weights from [13].

5.3 Main Results

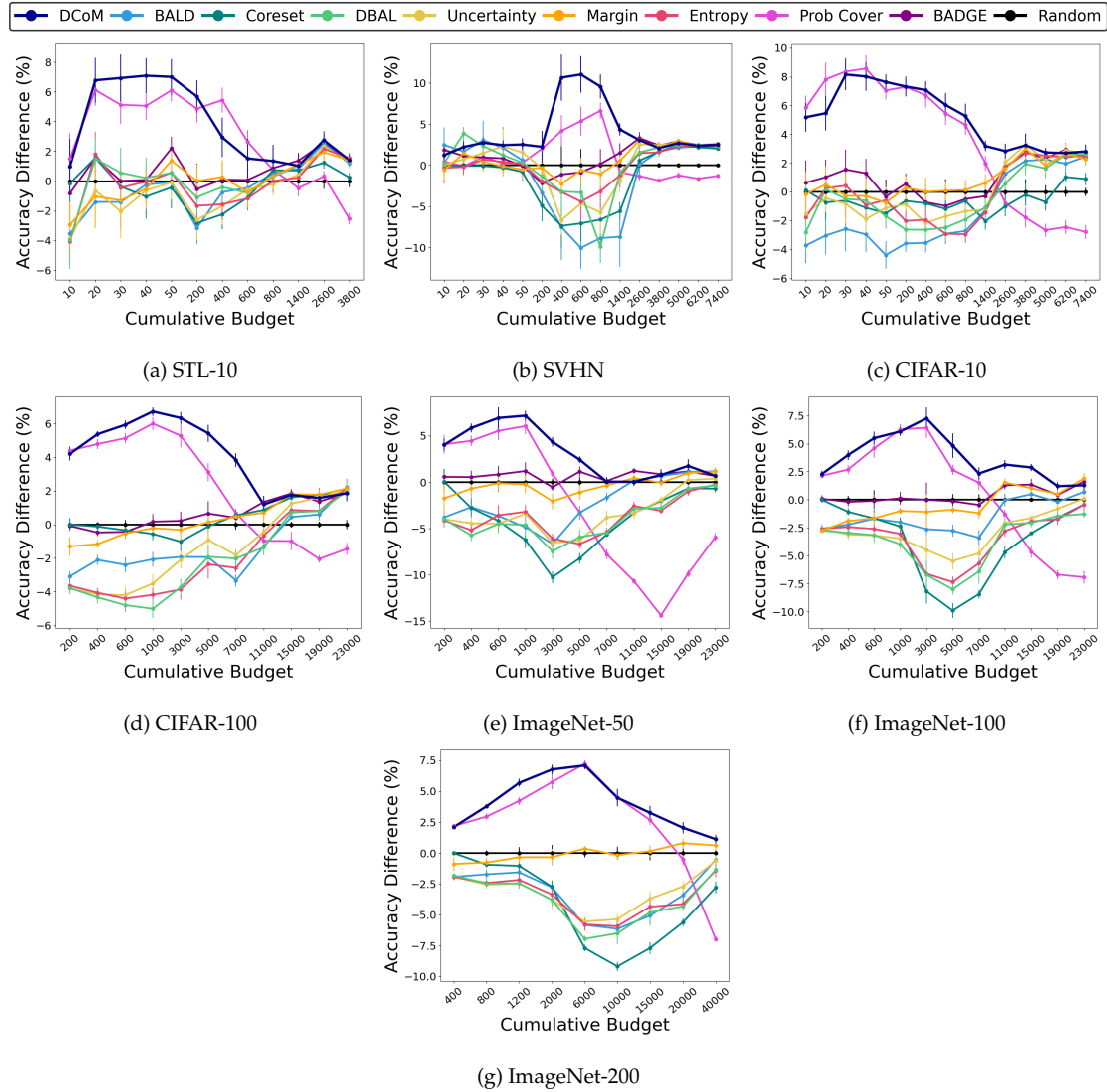


Figure 5.1: Mean accuracy difference between several AL algorithms and random queries. Positive mean difference indicates that the corresponding AL method is beneficial. The errors bars correspond to standard error, based on 5-10 repetitions (3 for the ImageNet subsets). While some methods perform well only in specific budget regimes, *DCoM* consistently achieves the best results across all budgets. Cumulative budget is evenly spaced for easy range comparison.

Fully supervised framework. We evaluate various AL methods by training a ResNet-18 model from scratch for each different AL strategy, using the labels of the queries selected by the respective method. In each AL iteration, a new model is trained from scratch using the updated labeled set \mathbb{L} . This process is repeated for several AL iterations. Fig. 5.1 shows the difference between the accuracy obtained by each AL method, and the accuracy obtained when training a similar network with a random query set of the same size (the baseline of no active learning). We show mean and the standard error over 5 – 10 repetitions (3 for the ImageNet’s subsets), for each AL iteration. The final accuracy of each method is shown in Tables 5.2-5.3 for the ImageNet-100 and CIFAR-100 datasets, as mean \pm Standard Error (STE). Additional datasets are shown in Section 5.4.

Table 5.2: Model final accuracy when varying the size of the labeled set \mathbb{L} (row) and the respective AL strategy (column), using the ImageNet-100 dataset

$ \mathbb{L} $	Random	Prob Cover	BADGE	BALD	Coreset	Uncertainty	Entropy	DBAL	Margin	DCom
200	4.58 \pm 0.13	6.72\pm0.07	4.67 \pm 0.16	1.99 \pm 0.22	4.58 \pm 0.13	2.01 \pm 0.06	1.99 \pm 0.10	1.83 \pm 0.06	1.84 \pm 0.12	6.87\pm0.18
400	6.61 \pm 0.17	9.30 \pm 0.23	6.42 \pm 0.23	4.39 \pm 0.39	5.53 \pm 0.16	3.50 \pm 0.32	4.17 \pm 0.33	3.65 \pm 0.25	4.71 \pm 0.10	10.59\pm0.26
600	8.11 \pm 0.42	12.71 \pm 0.42	8.01 \pm 0.20	6.42 \pm 0.24	6.45 \pm 0.37	4.93 \pm 0.29	5.51 \pm 0.33	4.96 \pm 0.22	6.46 \pm 0.12	13.61\pm0.15
1,000	11.43 \pm 0.17	17.71\pm0.27	11.56 \pm 0.38	9.42 \pm 0.19	9.04 \pm 0.66	7.97 \pm 0.82	8.39 \pm 0.18	7.45 \pm 0.18	10.43 \pm 0.11	17.53\pm0.23
3,000	27.59 \pm 0.76	34.01 \pm 0.15	27.58 \pm 0.72	24.96 \pm 0.23	19.41 \pm 0.32	23.10 \pm 0.60	20.97 \pm 0.33	20.88 \pm 0.52	26.51 \pm 0.79	34.85\pm0.23
5,000	39.25 \pm 0.26	41.91 \pm 0.20	39.13 \pm 0.06	36.50 \pm 0.25	29.35 \pm 0.39	33.78 \pm 0.46	31.89 \pm 0.24	31.25 \pm 0.21	38.37 \pm 0.10	44.09\pm0.83
7,000	46.88 \pm 0.25	48.41 \pm 0.42	46.42 \pm 0.41	43.49 \pm 0.42	38.45 \pm 0.12	42.09 \pm 0.54	41.19 \pm 0.17	40.47 \pm 0.83	45.69 \pm 0.26	49.22\pm0.29
11,000	55.31 \pm 0.26	54.01 \pm 0.15	56.61 \pm 0.11	55.29 \pm 0.32	50.63 \pm 0.33	53.25 \pm 0.31	52.52 \pm 0.26	53.09 \pm 0.29	56.86 \pm 0.10	58.45\pm0.15
15,000	61.69 \pm 0.09	57.04 \pm 0.43	63.05 \pm 0.42	62.19 \pm 0.21	58.71 \pm 0.05	60.07 \pm 0.07	59.76 \pm 0.22	59.64 \pm 0.30	62.71 \pm 0.27	64.57\pm0.22
19,000	66.39 \pm 0.26	59.69 \pm 0.16	66.82 \pm 0.33	66.19 \pm 0.19	64.77 \pm 0.32	65.59 \pm 0.27	64.65 \pm 0.23	64.93 \pm 0.23	66.89 \pm 0.28	67.60\pm0.08
23,000	68.92 \pm 0.15	61.99 \pm 0.41	70.54\pm0.21	69.62 \pm 0.32	68.49 \pm 0.06	69.03 \pm 0.20	68.47 \pm 0.50	67.65 \pm 0.14	70.81\pm0.35	70.21\pm0.32

Semi-supervised framework. We evaluate the effectiveness of 5 representative AL strategies with FlexMatch, a competitive semi-supervised learning method. We also evaluate our method, and the results of random sampling (no AL) as baseline. Fig. 5.2 shows the mean accuracy and STE based on 3 repetitions of FlexMatch, employing labeled sets generated by the different AL strategies. Note that in these experiments, since the query is selected from an unlabeled

Table 5.3: Same as Table 5.2, using the CIFAR-100 dataset.

$ \mathcal{L} $	Random	Prob Cover	BADGE	BALD	Coreset	Uncertainty	Entropy	DBAL	Margin	LDM-s	DCoM
200	6.39±0.19	10.79±0.08	6.31±0.09	3.30±0.11	6.39±0.19	2.59±0.05	2.74±0.17	2.61±0.18	5.09±0.37	-	10.59±0.19
400	8.74±0.07	13.53±0.22	8.27±0.14	6.63±0.23	8.62±0.17	4.59±0.23	4.68±0.23	4.41±0.30	7.58±0.17	-	14.11±0.11
600	10.73±0.13	15.86±0.16	10.29±0.27	8.33±0.31	10.39±0.25	6.52±0.38	6.32±0.27	5.94±0.28	10.20±0.16	-	16.66±0.11
1,000	13.49±0.18	19.49±0.20	13.66±0.26	11.42±0.26	12.94±0.21	9.98±0.38	9.31±0.22	8.47±0.35	13.28±0.19	-	20.21±0.07
3,000	24.02±0.27	29.31±0.42	24.24±0.28	22.10±0.11	23.00±0.28	21.93±0.26	20.15±0.33	20.29±0.23	23.71±0.31	-	30.36±0.08
5,000	31.52±0.39	34.64±0.15	32.18±0.33	29.58±0.38	31.41±0.26	30.60±0.40	29.15±0.46	29.60±0.28	31.66±0.23	-	36.95±0.10
7,000	37.69±0.16	38.36±0.13	38.09±0.10	34.36±0.20	38.24±0.16	35.87±0.19	35.11±0.11	35.67±0.35	38.16±0.17	31.85±0.00	41.52±0.25
11,000	45.66±0.22	44.69±0.20	47.00±0.16	44.37±0.28	46.53±0.18	45.23±0.20	45.01±0.17	44.31±0.24	46.37±0.36	40.88±0.01	46.86±0.28
15,000	50.57±0.09	49.58±0.46	52.42±0.17	51.02±0.09	52.19±0.29	51.82±0.16	51.43±0.21	51.30±0.35	52.37±0.19	46.59±0.01	52.33±0.22
19,000	54.95±0.09	52.90±0.14	56.32±0.07	55.55±0.27	56.72±0.31	56.55±0.21	55.76±0.31	55.75±0.14	56.73±0.24	49.41±0.01	56.53±0.07
23,000	57.74±0.14	56.29±0.21	59.60±0.16	59.96±0.35	59.59±0.22	59.79±0.23	59.91±0.08	59.72±0.21	59.88±0.21	52.48±0.00	59.60±0.33

set, the resulting labeled set \mathcal{L} may be unbalanced between the classes, which is especially harmful when the budget is small. This affects all the methods. As a result, the accuracy with random sampling may differ from reported results. Notably, *DCoM* demonstrates significant superiority over random sampling.

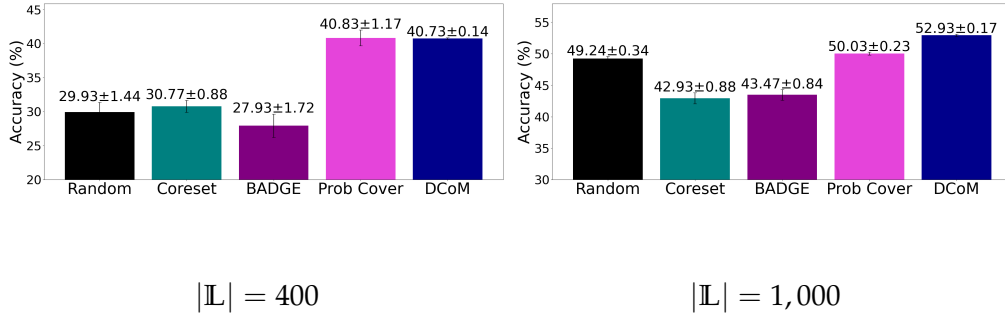


Figure 5.2: Comparison of AL strategies in a semi-supervised learning framework. Each bar represents the mean test accuracy over 3 repetitions of FlexMatch, trained using a total of 400 (left) and 1,000 (right) labeled examples on the CIFAR-100 dataset (an average of 4 and 10 labeled examples per class respectively). Three rounds of AL are considered, selecting 100 examples in the first round, 300 in the second, and 600 in the third.

5.4 Additional empirical results

5.4.1 δ_0 Initialization

To set the initial value δ_0 , we adopted the method outlined in [47], as detailed in Chapter 4. Fig. 5.3 displays the purity function across various δ values, along with the selected δ values for each dataset. As mentioned previously, the δ values for CIFAR-10 and CIFAR-100 are derived from [47].

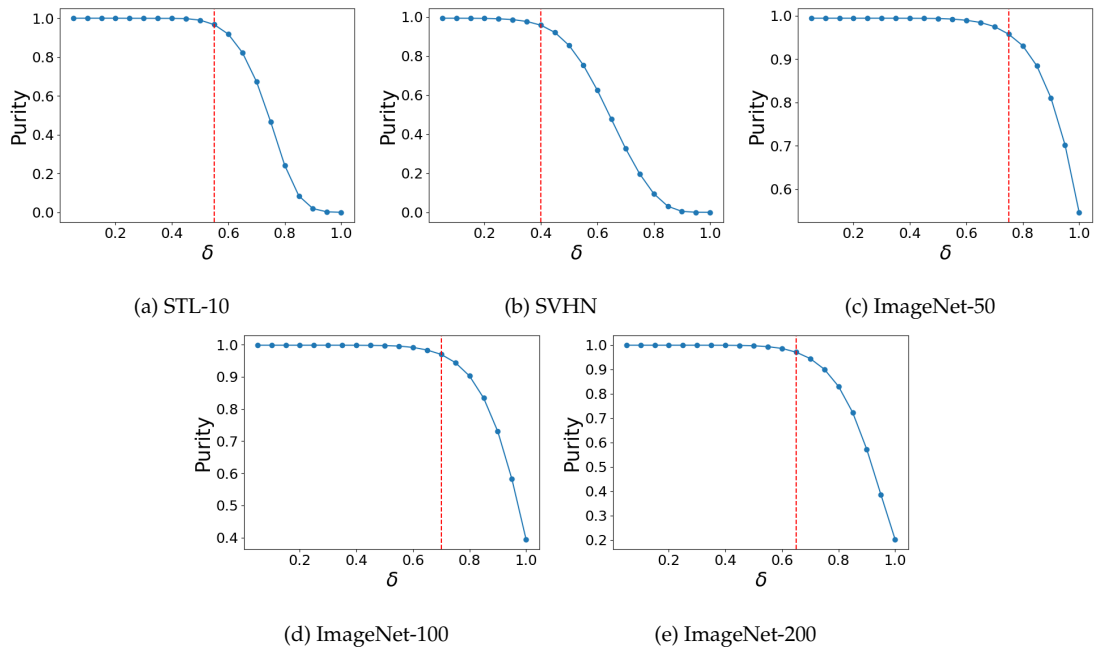


Figure 5.3: $\pi(\delta)$, estimated from the unlabeled data and using k-means algorithm for labeling. The dashed line indicates the highest δ , after which purity drops below $\alpha = 0.95$.

5.4.2 Δ distribution throughout the training process

The experiment in Fig. 5.4 illustrates the distribution of Δ throughout the active selection and training phases of *DCoM*. The significant standard deviation of Δ compared to the minor standard error underscores the significance of employing a dynamic method that assigns a distinct radius to each ball.

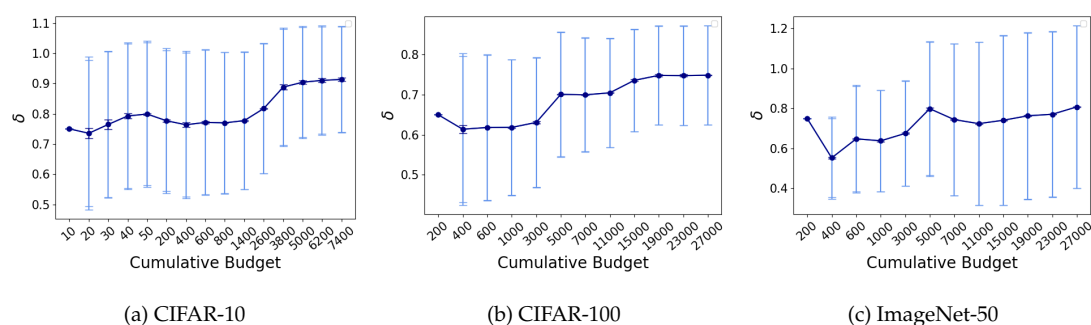


Figure 5.4: The Δ distribution through *DCoM* algorithm. Each plot displays the mean and standard error over 5-10 repetitions. The results indicate that as the emphasis shifts towards maximizing the purity of sample balls, rather than solely focusing on maximum coverage, the radii adapt accordingly. This, combined with the improved performance observed with *DCoM*, highlights the necessity of the dynamic algorithm in enhancing the effectiveness of active sampling.

5.4.3 Dynamic algorithm coverage

The analysis presented in Fig. 5.5 demonstrates a consistent behavior of probability coverage across different datasets during the *DCoM* process. This indicates that using probability coverage as a measure of progress consistently proves to be a wise decision.

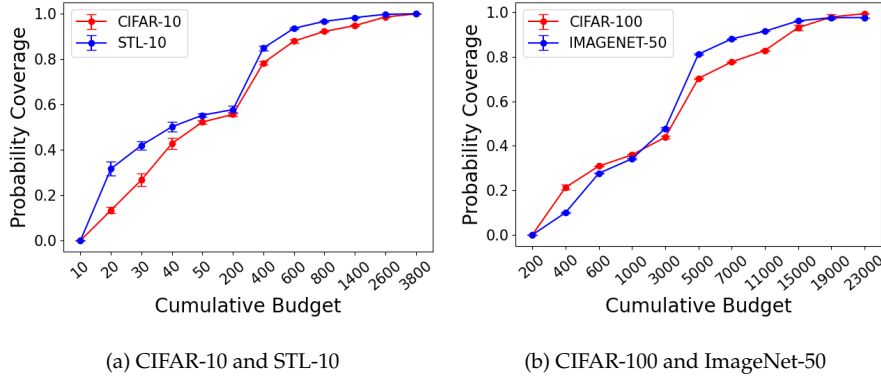


Figure 5.5: Comparison between $P(C(\mathbb{L}, \Delta))$ values during *DCoM* sampling process, across several datasets. Each plot displays the mean and standard error over 3 – 10 repetitions. These figures demonstrates a consistent behavior of probability coverage.

5.4.4 Decomposing the algorithm

The objective function in (3.1) includes the linear combination of two objective functions, weighed by competence score $S_{\mathcal{L}}(\mathbb{L}, \Delta)$ from (3.6). The general form of this objective function is an important part of our method, and we therefore We evaluate the contribution of each component in objective function (3.1) separately. Results are shown in Fig. 5.6, wherein *DPC* denotes the Dynamic Probability Coverage, corresponding to $\mathcal{O}_{low}(x)$ in *DCoM*, and *Margin* denotes *DCoM*'s choice for $\mathcal{O}_{high}(x)$. Clearly, the weighted objective function of *DCoM* yields superior results over its components across all labeled set sizes $|\mathbb{L}|$.

In Fig. 5.7 you can see the same ablation but using entropy as $\mathcal{O}_{high}(x)$ instead of margin.

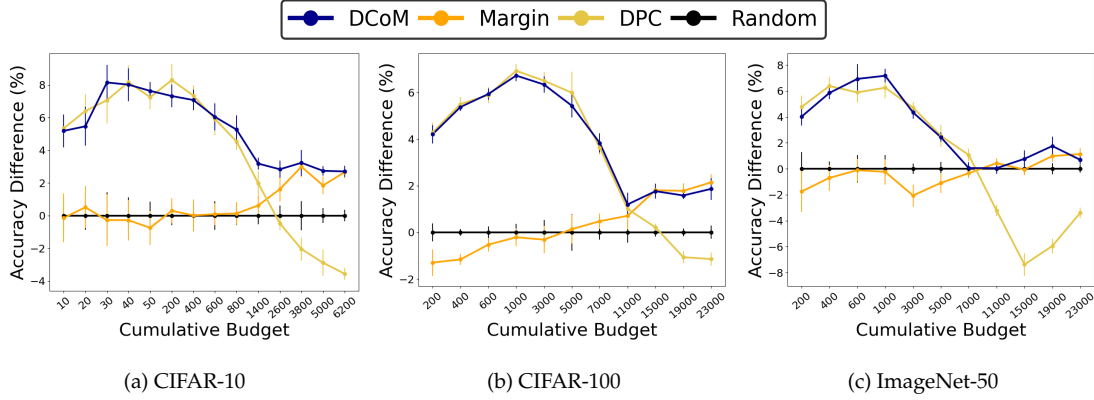


Figure 5.6: Performance evaluation (mean accuracy and STE) when optimizing each component of objective function (3.1) separately and together. DPC denotes the Dynamic Probability Coverage, corresponding to $\mathcal{O}_{low}(x)$ in $DCoM$. Margin is $DCoM$'s choice for $\mathcal{O}_{high}(x)$. Each plot displays the mean and standard error over 5 – 10 repetitions.

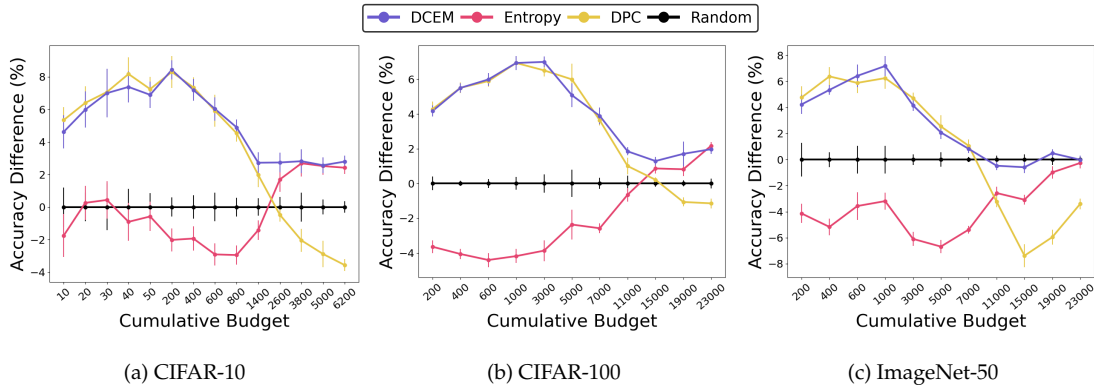


Figure 5.7: Assessment of each algorithm's objective function individually across multiple datasets. DPC stands for Dynamic Probability Coverage, corresponding to $\mathcal{O}_{low}(x)$ in $DCoM$. $DCEM$ represents $DCoM$ with entropy as $\mathcal{O}_{high}(x)$ instead of margin. Each plot displays the mean and standard error over 5 – 10 repetitions. These graphs demonstrate how the weighted objective function consistently produces superior results across all labeled set sizes $|\mathbb{L}|$.

5.4.5 Different embedding space

As discussed in Section 4.1, our approach relies on the existence of a suitable embedding space, where distance is indicative of semantic dissimilarity. We now repeat the basic fully-supervised experiments while varying the embedding employed by *DCoM*, considering various popular self-supervised representations. Results are reported in Fig. 5.8, showing that *DCoM* consistently delivers the best performance regardless of the specific embedding used.

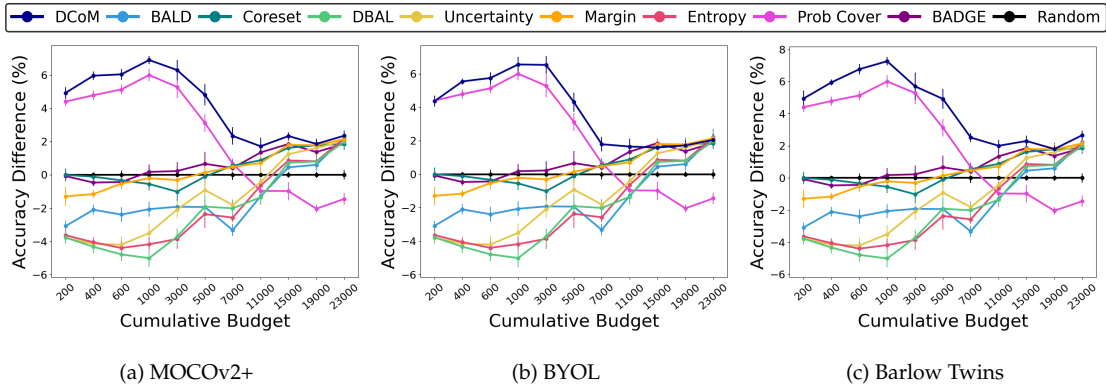


Figure 5.8: Performance over CIFAR-100, using different embedding spaces for *DCoM* and *ProbCover*, see details in the caption of Fig. 5.1. Clearly, *DCoM* consistently achieves the best results.

5.4.6 Different uncertainty-based methods.

We now evaluate our choice for $\mathcal{O}_{high}(x)$ in objective function (3.1), investigating the alternative methods of min margin, entropy of the min max activation, and uncertainty (see details in Section 5.1). Additionally, we use a score reminiscent of the AL method BADGE [16] and termed *Gradient norm*, which computes the norm of the gradient between the learner’s last two layers. Once again, the results of *DCoM* show robustness to this choice, with insignificant differences

between the different uncertainty scores, see Fig. 5.10. Consequently, we prioritize computational efficiency and opt for the fastest-to-compute score, which is min-margin.

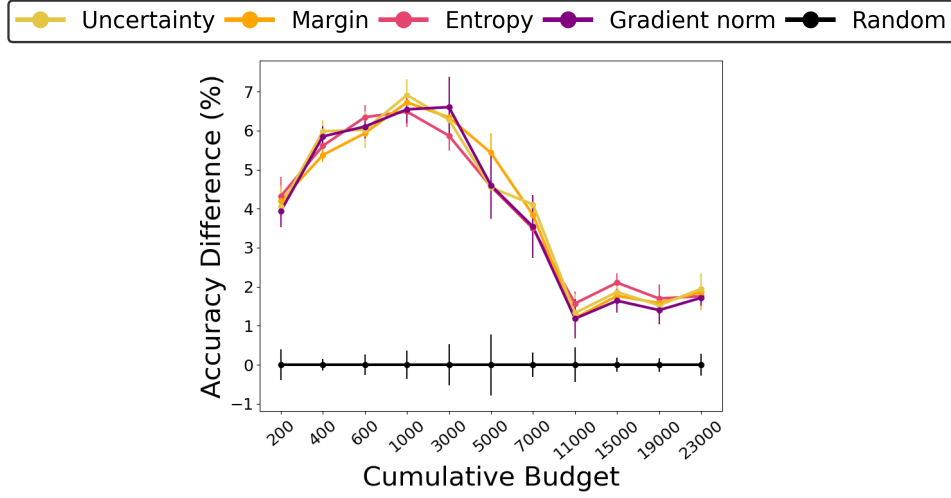


Table 5.4: Model accuracy for different sizes of labeled set \mathbb{L} and several uncertainty-based methods for $\mathbb{F}(x)$

$ \mathbb{L} $	Random	Entropy	Margin	Uncertainty	Gradient norm
200	6.39±0.19	10.71±0.30	10.59±0.19	10.44±0.26	10.33±0.21
400	8.74±0.07	14.35±0.24	14.11±0.11	14.72±0.20	14.59±0.19
600	10.73±0.13	17.07±0.19	16.66±0.11	16.75±0.34	16.83±0.17
1,000	13.49±0.18	19.98±0.22	20.21±0.07	20.40±0.21	20.03±0.17
3,000	24.02±0.27	29.89±0.11	30.36±0.08	30.30±0.11	30.62±0.52
5,000	31.52±0.39	36.08±0.13	36.95±0.10	36.06±0.18	36.12±0.46
7,000	37.69±0.16	41.18±0.25	41.52±0.25	41.80±0.08	41.23±0.65
11,000	45.66±0.22	47.23±0.09	46.86±0.28	46.99±0.27	46.84±0.28
15,000	50.57±0.09	52.67±0.15	52.33±0.22	52.43±0.14	52.21±0.22
19,000	54.95±0.09	56.64±0.27	56.53±0.07	56.47±0.06	56.35±0.28
23,000	57.74±0.14	59.50±0.18	59.60±0.33	59.68±0.24	59.45±0.06

Figure 5.10: Similar to Fig. 5.1 and Tables 5.2-5.3, using only *DCoM* while leveraging 4 different variants for its underlying uncertainty score. Each variation incorporates a dynamic algorithm weighted with a different uncertainty-based method. Performance is assessed by the accuracy difference between each variation and no active learning (random sampling). Each plot corresponds to 3 – 5 repetitions using the CIFAR-100 dataset. These findings suggest that when the algorithm begins with optimization on $\mathcal{O}_{low}(x)$, the performance differences between the variants are small.

5.4.7 Comparison per dataset

Tables 5.6-5.6 present the empirical accuracy values of several datasets using different active learning algorithms. In each active step, a new learner is trained from scratch over all available labeled data, and the results are presented as the [mean \pm STE] over 5 – 10 repetitions (3 for ImageNet subsets). As mentioned earlier, the results for LDM-s algorithm were acquired from the study by [8]. Their lack of code provision suggests discrepancies in the running setup. Nonetheless, the disparity between their method and random selection in their setup is smaller than that observed in ours, implying the superiority of our method. The table includes the results for all datasets, with the name of each dataset below it.

Table 5.5: Model accuracy for different sizes of labeled set \mathbb{L} and AL strategies

$ \mathbb{L} $	Random	Prob Cover	BADGE	BALD	Coreset	Uncertainty	Entropy	DBAL	Margin	LDM-s	DCoM
200	6.39±0.19	10.79±0.08	6.31±0.09	3.30±0.11	6.39±0.19	2.59±0.05	2.74±0.17	2.61±0.18	5.09±0.37	-	10.59±0.19
400	8.74±0.07	13.53±0.22	8.27±0.14	6.63±0.23	8.62±0.17	4.59±0.23	4.68±0.23	4.41±0.30	7.58±0.17	-	14.11±0.11
600	10.73±0.13	15.86±0.16	10.29±0.27	8.33±0.31	10.39±0.25	6.52±0.38	6.32±0.27	5.94±0.28	10.20±0.16	-	16.66±0.11
1,000	13.49±0.18	19.49±0.20	13.66±0.26	11.42±0.26	12.94±0.21	9.98±0.38	9.31±0.22	8.47±0.35	13.28±0.19	-	20.21±0.07
3,000	24.02±0.27	29.31±0.42	24.24±0.28	22.10±0.11	23.00±0.28	21.93±0.26	20.15±0.33	20.29±0.23	23.71±0.31	-	30.36±0.08
5,000	31.52±0.39	34.64±0.15	32.18±0.33	29.58±0.38	31.41±0.26	30.60±0.40	29.15±0.46	29.60±0.28	31.66±0.23	-	36.95±0.10
7,000	37.69±0.16	38.36±0.13	38.09±0.10	34.36±0.20	38.24±0.16	35.87±0.19	35.11±0.11	35.67±0.35	38.16±0.17	31.85±0.00	41.52±0.25
11,000	45.66±0.22	44.69±0.20	47.00±0.16	44.37±0.28	46.53±0.18	45.23±0.20	45.01±0.17	44.31±0.24	46.37±0.36	40.88±0.01	46.86±0.28
15,000	50.57±0.09	49.58±0.46	52.42±0.17	51.02±0.09	52.19±0.29	51.82±0.16	51.43±0.21	51.30±0.35	52.37±0.19	46.59±0.01	52.33±0.22
19,000	54.95±0.09	52.90±0.14	56.32±0.07	55.55±0.27	56.72±0.31	56.55±0.21	55.76±0.31	55.75±0.14	56.73±0.24	49.41±0.01	56.53±0.07
23,000	57.74±0.14	56.29±0.21	59.60±0.16	59.96±0.35	59.59±0.22	59.79±0.23	59.91±0.08	59.72±0.21	59.88±0.21	52.48±0.00	59.60±0.33

CIFAR-100 dataset

$ \mathbb{L} $	Random	Prob Cover	BADGE	BALD	Coreset	Uncertainty	Entropy	DBAL	Margin	DCoM
200	8.25±0.65	12.33±0.34	8.82±0.18	4.46±0.31	8.31±0.52	4.26±0.26	4.11±0.09	4.26±0.24	6.50±0.94	12.27±0.04
400	12.18±0.29	16.64±0.29	12.73±0.42	9.49±0.84	9.40±0.59	7.74±0.41	7.01±0.37	6.45±0.33	11.48±0.72	18.03±0.21
600	14.85±0.53	20.38±0.47	15.69±0.38	11.21±0.41	10.68±0.35	10.38±0.27	11.28±0.54	10.34±0.46	14.74±0.35	21.77±0.62
1,000	19.71±0.53	25.76±0.34	20.91±0.43	14.85±0.35	13.47±0.32	16.30±0.60	16.52±0.12	15.10±0.65	19.47±0.45	26.88±0.00
3,000	39.06±0.21	39.98±0.23	38.51±0.15	32.40±0.40	28.80±0.45	32.52±0.63	32.97±0.34	31.60±0.60	37.00±0.65	43.40±0.29
5,000	48.33±0.27	45.08±0.16	49.47±0.26	45.04±0.47	40.10±0.31	42.05±0.29	41.65±0.22	42.41±0.52	47.24±0.50	50.76±0.08
7,000	55.12±0.09	47.34±0.44	55.25±0.34	53.48±0.38	49.48±0.19	51.30±0.68	49.74±0.21	49.67±0.46	54.78±0.29	55.17±0.34
11,000	63.46±0.15	52.79±0.17	64.68±0.15	63.65±0.32	60.13±0.11	60.10±0.50	60.89±0.34	60.58±0.25	63.90±0.22	63.51±0.31
15,000	69.42±0.20	55.04±0.06	70.27±0.25	70.08±0.21	67.38±0.17	67.49±0.19	66.34±0.16	66.65±0.23	69.36±0.24	70.19±0.45
19,000	72.54±0.21	62.66±0.19	73.70±0.29	73.63±0.21	71.87±0.14	72.78±0.41	71.55±0.27	71.85±0.17	73.52±0.23	74.29±0.50
23,000	75.99±0.16	70.06±0.27	76.65±0.11	77.17±0.06	75.30±0.24	76.37±0.29	75.73±0.25	75.64±0.27	77.12±0.30	76.68±0.20

ImageNet-50 dataset

$ \mathbb{L} $	Random	Prob Cover	BADGE	BALD	Coreset	Uncertainty	Entropy	DBAL	Margin	DCoM
200	4.58±0.13	6.72±0.07	4.67±0.16	1.99±0.22	4.58±0.13	2.01±0.06	1.99±0.10	1.83±0.06	1.84±0.12	6.87±0.18
400	6.61±0.17	9.30±0.23	6.42±0.23	4.39±0.39	5.53±0.16	3.50±0.32	4.17±0.33	3.65±0.25	4.71±0.10	10.59±0.26
600	8.11±0.42	12.71±0.42	8.01±0.20	6.42±0.24	6.45±0.37	4.93±0.29	5.51±0.33	4.96±0.22	6.46±0.12	13.61±0.15
1,000	11.43±0.17	17.71±0.27	11.56±0.38	9.42±0.19	9.04±0.66	7.97±0.82	8.39±0.18	7.45±0.18	10.43±0.11	17.53±0.23
3,000	27.59±0.76	34.01±0.15	27.58±0.72	24.96±0.23	19.41±0.32	23.10±0.60	20.97±0.33	20.88±0.52	26.51±0.79	34.85±0.23
5,000	39.25±0.26	41.91±0.20	39.13±0.06	36.50±0.25	29.35±0.39	33.78±0.46	31.89±0.24	31.25±0.21	38.37±0.10	44.09±0.83
7,000	46.88±0.25	48.41±0.42	46.42±0.41	43.49±0.42	38.45±0.12	42.09±0.54	41.19±0.17	40.47±0.83	45.69±0.26	49.22±0.29
11,000	55.31±0.26	54.01±0.15	56.61±0.11	55.29±0.32	50.63±0.33	53.25±0.31	52.52±0.26	53.09±0.29	56.86±0.10	58.45±0.15
15,000	61.69±0.09	57.04±0.43	63.05±0.42	62.19±0.21	58.71±0.05	60.07±0.07	59.76±0.22	59.64±0.30	62.71±0.27	64.57±0.22
19,000	66.39±0.26	59.69±0.16	66.82±0.33	66.19±0.19	64.77±0.32	65.59±0.27	64.65±0.23	64.93±0.23	66.89±0.28	67.60±0.08
23,000	68.92±0.15	61.99±0.41	70.54±0.21	69.62±0.32	68.49±0.06	69.03±0.20	68.47±0.50	67.65±0.14	70.81±0.35	70.21±0.32

ImageNet-100 dataset

$ \mathbb{L} $	Random	Prob Cover	BALD	Coreset	Uncertainty	Entropy	DBAL	Margin	DCoM
400	3.10±0.07	5.28±0.08	1.18±0.21	3.10±0.07	1.19±0.07	1.17±0.13	1.29±0.06	2.22±0.50	5.20±0.02
800	4.61±0.10	7.56±0.15	2.90±0.29	3.68±0.12	2.06±0.24	2.20±0.19	2.15±0.20	3.85±0.34	8.39±0.07
1,200	5.88±0.25	10.10±0.08	4.32±0.33	4.84±0.14	3.42±0.19	3.71±0.26	3.42±0.16	5.54±0.23	11.56±0.07
2,000	9.51±0.33	15.25±0.26	6.73±0.26	6.78±0.15	5.72±0.41	6.15±0.04	5.72±0.15	9.17±0.34	16.27±0.08
6,000	25.67±0.16	32.89±0.12	19.85±0.26	17.98±0.10	20.12±0.11	19.89±0.35	18.72±0.08	26.03±0.08	32.76±0.13
10,000	37.49±0.26	42.00±0.16	31.36±0.46	28.31±0.10	32.13±0.16	31.57±0.28	31.00±0.59	37.33±0.08	41.97±0.44
15,000	46.22±0.30	48.89±0.09	41.16±0.49	38.53±0.18	42.53±0.29	41.88±0.31	41.42±0.13	46.37±0.17	49.47±0.27
20,000	52.20±0.19	51.70±0.32	48.79±0.58	46.59±0.15	49.50±0.25	48.07±0.36	47.89±0.13	53.01±0.19	54.25±0.26
40,000	64.57±0.17	57.60±0.05	64.02±0.37	61.78±0.30	63.94±0.27	63.14±0.37	63.21±0.13	65.19±0.24	65.69±0.17

ImageNet-200 dataset

Table 5.6: Model accuracy for different sizes of labeled set \mathbb{L} and AL strategies

$ \mathbb{L} $	Random	Prob Cover	BADGE	BALD	Coreset	Uncertainty	Entropy	DBAL	Margin	LDM-s	DCoM
10	14.53±0.60	20.39±0.23	15.17±0.94	10.81±0.61	14.65±0.74	12.99±0.77	12.78±0.72	11.74±0.69	14.39±0.88	-	19.72±0.40
20	16.82±0.44	24.64±0.73	17.85±0.77	13.79±0.89	16.09±0.74	16.38±0.52	17.07±0.62	17.00±0.49	17.33±0.85	-	22.28±0.75
30	18.82±0.71	27.19±0.25	20.38±0.67	16.25±0.83	18.22±0.93	18.00±0.68	19.26±0.44	18.34±0.50	18.55±0.87	-	26.98±0.35
40	20.89±0.55	29.46±0.38	22.19±0.45	17.93±0.67	19.79±0.61	19.00±0.70	19.98±0.60	20.30±0.35	20.61±0.68	-	28.91±0.46
50	22.75±0.42	29.80±0.21	22.41±0.35	18.36±0.54	21.26±0.63	21.63±0.70	22.17±0.48	21.06±0.45	22.00±0.61	-	30.39±0.12
200	31.53±0.29	38.83±0.18	32.10±0.42	27.95±0.29	30.91±0.76	30.69±0.27	29.51±0.42	28.91±0.50	31.83±0.48	-	38.86±0.40
400	38.64±0.36	45.36±0.51	37.92±0.37	35.10±0.33	37.85±0.70	36.48±0.46	36.71±0.38	36.01±0.32	38.65±0.62	-	45.72±0.27
600	43.66±0.44	49.13±0.36	42.69±0.35	40.75±0.29	42.48±0.39	41.96±0.24	40.75±0.23	41.19±0.53	43.75±0.22	-	49.71±0.39
800	46.84±0.29	51.49±0.37	46.34±0.32	44.12±0.49	46.21±0.25	45.50±0.17	43.89±0.31	44.93±0.42	46.97±0.40	-	52.10±0.58
1,400	54.35±0.27	56.33±0.18	54.06±0.42	53.08±0.34	52.31±0.31	53.13±0.28	52.93±0.34	53.28±0.50	54.97±0.35	50.30±0.00	57.54±0.10
2,600	62.93±0.31	62.16±0.18	64.67±0.31	64.19±0.24	61.94±0.40	64.98±0.59	64.62±0.46	63.53±0.44	64.54±0.45	57.01±0.01	65.76±0.21
3,800	68.61±0.44	66.83±0.25	71.36±0.32	70.77±0.53	68.41±0.34	71.90±0.36	71.30±0.36	70.55±0.25	71.60±0.21	61.30±0.01	71.84±0.36
5,000	73.61±0.23	70.95±0.23	75.93±0.21	75.86±0.27	72.92±0.36	75.82±0.31	76.13±0.31	75.24±0.47	75.46±0.31	64.09±0.00	76.36±0.05
6,200	75.95±0.18	73.51±0.30	78.90±0.27	77.92±0.36	77.00±0.19	78.84±0.27	78.37±0.20	78.54±0.26	78.62±0.22	65.79±0.00	78.66±0.16
7,400	78.49±0.14	75.71±0.35	80.88±0.22	80.94±0.16	79.39±0.30	81.19±0.15	81.02±0.19	81.10±0.24	80.76±0.29	67.28±0.01	81.30±0.34

CIFAR-10 dataset

$ \mathbb{L} $	Random	Prob Cover	BADGE	BALD	Coreset	Uncertainty	Entropy	DBAL	Margin	DCoM
10	15.73±0.90	17.26±0.78	14.92±0.98	12.22±0.91	15.58±1.03	11.88±0.83	11.67±0.48	11.79±1.06	12.81±0.65	16.72±0.94
20	16.68±0.77	22.81±0.28	18.24±0.46	15.28±0.98	18.25±0.97	16.05±0.94	18.48±0.67	18.19±0.63	15.65±1.35	23.47±0.75
30	19.78±0.79	24.91±0.49	19.80±0.38	18.37±1.06	19.41±0.92	17.75±1.05	19.33±0.74	20.35±0.90	18.49±1.16	26.72±0.79
40	21.49±0.48	26.56±0.48	21.57±0.48	21.25±0.95	20.45±1.00	20.86±0.77	21.45±0.74	21.70±0.88	20.97±0.71	28.58±0.69
50	22.19±0.33	28.30±0.42	24.40±0.45	22.13±0.56	21.76±1.07	22.11±0.81	22.75±0.60	22.76±0.91	23.57±0.59	29.20±0.87
200	36.04±0.59	40.91±0.33	35.50±0.57	32.89±0.50	33.22±0.55	33.44±0.76	34.38±0.58	34.95±0.46	36.05±0.53	41.74±0.49
400	44.07±0.60	49.52±0.23	44.18±0.54	43.33±0.45	41.83±0.39	42.39±0.64	42.51±0.46	43.67±0.47	44.35±0.52	47.02±0.73
600	50.02±0.44	52.65±0.35	50.10±0.44	49.56±0.32	48.91±0.32	49.46±0.48	48.85±0.38	49.23±0.49	49.27±0.59	51.54±0.86
800	53.69±0.36	54.45±0.56	54.57±0.63	54.20±0.51	54.42±0.21	53.48±0.38	53.70±0.38	53.98±0.31	54.05±0.56	55.05±0.71
1,400	62.00±0.26	61.54±0.30	63.38±0.25	62.82±0.51	62.72±0.42	62.42±0.27	62.24±0.23	63.04±0.23	63.11±0.25	63.04±0.57
2,600	70.18±0.27	70.50±0.08	72.76±0.19	72.72±0.26	71.41±0.13	72.53±0.23	72.35±0.18	72.97±0.18	72.12±0.40	72.93±0.33
3,800	75.51±0.21	72.97±0.17	76.75±0.15	76.69±0.11	75.75±0.09	76.80±0.21	77.05±0.15	76.85±0.09	76.91±0.10	76.92±0.18

STL-10 dataset

$ \mathbb{L} $	Random	Prob Cover	BADGE	BALD	Coreset	Uncertainty	Entropy	DBAL	Margin	LDM-s	DCoM
10	11.36±0.50	11.80±0.08	13.26±0.79	13.84±1.64	11.36±0.50	11.41±0.85	11.06±1.38	11.66±0.56	10.80±0.66	-	12.62±0.60
20	11.05±0.41	10.96±0.19	12.19±0.67	12.74±1.26	10.99±0.59	11.81±0.84	10.86±0.58	14.94±0.36	12.50±0.95	-	13.30±0.26
30	11.16±0.36	12.19±0.52	12.12±0.51	14.22±2.01	11.10±0.29	12.66±1.28	11.86±0.65	13.50±0.46	11.56±0.79	-	13.91±0.32
40	11.85±0.46	12.11±0.20	12.71±0.70	13.98±2.06	11.57±0.57	14.16±1.63	12.28±0.47	13.10±0.41	11.76±0.75	-	14.32±0.14
50	12.55±0.49	12.86±0.36	12.47±0.89	13.18±2.00	11.76±0.38	14.12±1.00	12.72±0.54	12.88±0.54	12.04±0.68	-	15.09±0.25
200	22.32±0.84	24.34±0.44	20.18±1.09	19.00±1.66	17.42±1.03	21.99±1.11	20.38±1.08	20.98±0.99	22.03±0.87	-	24.58±1.10
400	36.24±1.24	40.42±0.67	35.13±0.40	28.74±2.78	28.88±1.47	29.58±0.55	33.01±1.36	33.06±1.82	34.01±0.18	-	46.90±1.56
600	47.29±0.98	52.67±0.75	46.46±0.87	37.26±1.56	40.24±1.24	42.65±0.68	42.89±1.38	43.99±0.46	46.69±0.50	-	58.34±1.25
800	56.32±0.50	62.96±0.51	56.48±1.33	47.47±2.46	49.74±1.42	50.62±2.03	53.17±1.64	46.45±1.14	55.27±1.05	-	65.90±0.97
1,400	74.00±0.38	73.16±0.21	75.53±0.77	65.33±3.30	68.44±0.69	72.13±0.98	72.81±1.09	73.14±0.69	74.53±0.74	79.02±0.01	78.36±0.36
2,600	83.24±0.48	81.94±0.08	86.56±0.24	83.33±0.96	83.83±0.39	85.80±0.31	84.72±0.54	84.83±0.59	86.33±0.13	83.76±0.00	86.31±0.26
3,800	87.18±0.14	85.34±0.11	89.61±0.25	88.91±0.54	88.86±0.14	89.64±0.09	88.79±0.14	89.36±0.15	89.34±0.18	86.08±0.00	89.28±0.13
5,000	88.79±0.15	87.58±0.20	91.72±0.15	90.93±0.14	91.24±0.17	91.55±0.04	91.13±0.11	91.28±0.12	91.75±0.17	87.58±0.00	91.50±0.08
6,200	90.16±0.04	88.55±0.10	92.64±0.09	92.46±0.04	92.35±0.03	92.63±0.08	92.56±0.16	92.65±0.10	92.55±0.12	88.64±0.00	92.54±0.12
7,400	90.99±0.11	89.71±0.08	93.56±0.05	93.26±0.10	93.02±0.14	93.33±0.05	93.38±0.07	93.38±0.11	93.43±0.13	89.56±0.00	93.47±0.09

SVHN dataset

5.5 Hyper-parameters exploring

As described before, *DCoM* has 3 hyper-parameters: a , k and τ . When running the main experiments described in Section 5.3, we maintained consistency by employing identical parameters for STL-10, SVHN, and CIFAR-10, as well as for CIFAR-100 and the subsets of ImageNet. This approach helps to demonstrate that the choice of the hyper-parameters is not critical and does not significantly affect the results. Here, we conduct experiments with *DCoM* using STL-10 and CIFAR-100 datasets and several values for a and k from the definition of the competence score $S_{\mathcal{L}}(\mathbb{L}, \Delta)$ in (3.6). We observe that while certain choices may be slightly better than others, all choices yield results that are very close to each other, and *DCoM* consistently outperforms or matches the performance of previous methods.

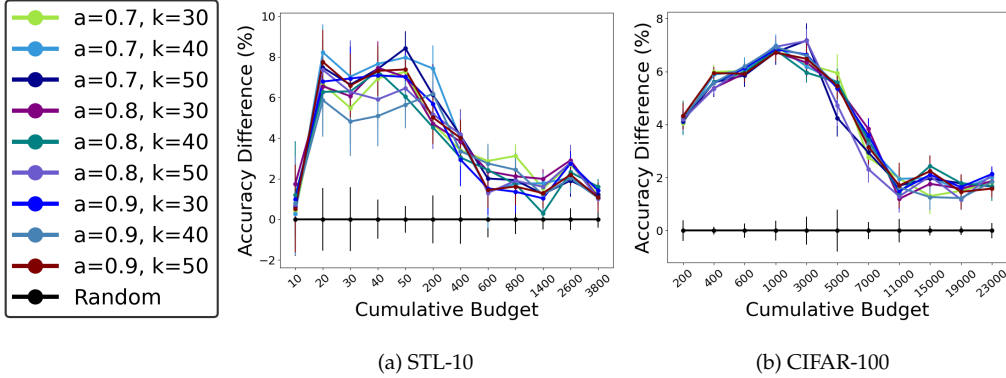


Figure 5.11: Evaluation of *DCoM* across different settings of the logistic function hyper-parameters a and k . Each plot displays the mean and standard error over 3 – 5 repetitions. The findings indicate that fine-tuning these parameters has negligible effects on performance.

6 Constrained K-Means Variation

6.1 Background

K-means is a widely utilized clustering algorithm in machine learning, designed to partition a dataset into K distinct, non-overlapping clusters. The algorithm iteratively assigns each data point to one of the K clusters, aiming to minimize the within-cluster variance, typically using the Euclidean distance as the metric. The process starts with K initial centroids and alternates between updating the cluster assignments and recalculating the centroids until convergence.

During the years, several variations of the basic K-means algorithm have been developed to address specific limitations and enhance performance. one of them is **COP-Kmeans**. COP-Kmeans incorporates must-link and cannot-link constraints to guide the clustering process. Must-link constraints specify that certain pairs of data points must be in the same cluster, while cannot-link constraints specify that certain pairs of data points must not be in the same cluster. These constraints are particularly useful in scenarios where prior domain knowledge about the relationships between data points is available, allowing for more accurate and meaningful clustering results. COP-Kmeans modifies the assignment step of the traditional Kmeans algorithm to ensure that these constraints are satisfied, thereby improving the clustering quality in constrained

environments.

Wagstaff et al. [41] introduces the initial version of the COP-Kmeans algorithm. This straightforward implementation involves defining the initial centers, $C = \{c_1, c_2, \dots, c_k\}$, iterating through each data point, and assigning it to the nearest cluster that satisfies the given constraints.

Huang et al. [25] presents an enhancement to the classic algorithm. This variation uses methods like k-means++ [3] to define the initial centers, $C = \{c_1, c_2, \dots, c_k\}$, and employs must-link constraints to compute the mass center of each constrained set. Each mass center is assigned to the nearest center from C . The algorithm then calculates the distance from each datapoint to the centers in from C , assigning points to clusters while respecting cannot-link constraints. Finally, it assigns each point in the cannot-link sets to the cluster that minimizes the total weight matching.

6.2 COPAL-Kmeans: Kmeans Variation For Active Learning

We use the available labels from \mathbb{L} as constraints in our implementation of the new variation. Initially, we assign all the points not in \mathbb{L} according to the minimum distance principle of the classic algorithm.

Next, we create sets of datapoints from \mathbb{L} that we already know belong to the same class. We then aim to match each known set to the best cluster center by calculating the distances between each mass center and the cluster centers. These distances are stored in matrix D .

Note that the points in \mathbb{L} are ideally subject to both must-link and cannot-link

constraints from the original variation. Finally, we use the matrix D to fit each set's mass center from \mathbb{L} to the corresponding cluster centroid from C in a way that minimizes the total linear sum of distances between mass centers and their assigned cluster centroids. (see Alg 3 below for pseudo-code).

6.3 Combining COPAL-Kmeans in $D\text{CoM}$

As you can read in Chapter 4, $D\text{CoM}$ includes a step of δ -expansion for new samples. We modify this step by limiting the δ -expansion with a purity threshold τ' that is based on the COPAL-Kmeans variation rather than the centroid label. Instead of computing a purity threshold $\tau = m \cdot P(C(\mathbb{L} \setminus Q, \Delta)) + b$, where m and b are hyperparameters, we use this new purity threshold τ' . This approach reduces the number of hyperparameters. (See Alg 4 below for pseudo-code).

Algorithm 3 Constrained K-Means Clustering using Domain Information

Input: unlabeled pool \mathbb{U} , labeled pool \mathbb{L} , a positive integer k

Output: A collection of k clusters $C = \{C_1, C_2, \dots, C_k\}$

Use the k-means++ algorithm to select c_1, c_2, \dots, c_k as the initial cluster centers

Set $C_i := \emptyset$, for $1 \leq i \leq k$

$\mathbb{X} \leftarrow \mathbb{U} \cup \mathbb{L}$

$S \leftarrow$ disjoint sets from \mathbb{L} where each set refers to examples with the same label

while cluster centers change **do**

for $x_i \in \mathbb{U}$ **do**

 Compute the distance m_{ij} between point x_i and each cluster center c_j ;

 Assign x_i to its nearest c_j ;

end for

for $s_i \in S$ **do**

 Compute mass center sample of the set s_i ;

 Compute the distance d_{ij} between the current mass center and each cluster center c_j ;

end for

for $s_i \in S$ **do**

 Assign each point of the set s_i to the appropriate cluster via minimum weight matching of the matrix d ;

end for

for $i = 1$ to k **do**

 Compute the cluster center c_i of C_i ;

end for

end while

return $C = \{C_1, C_2, \dots, C_k\}$

Algorithm 4 δ -expansion with COPAL-Kmeans

Input: unlabeled pool \mathbb{U} , labeled pool \mathbb{L} , query pool Q , maximal ball size δ_{\max} , list of ball sizes Δ for $\mathbb{L} \setminus Q$, trained model \mathbb{M} from the current iteration and $P(C(\mathbb{L} \setminus Q, \Delta))$ from **Active sampling**, τ' purity threshold for COPAL-Kmeans
Output: updated list of ball sizes Δ

$\hat{\mathbb{Y}}_{\text{COPAL}} \leftarrow$ The predicted cluster for each $x \in \mathbb{U}$ the COPAL-Kmeans algorithm
for $v \in Q$ **do**
 $\delta_{\text{opt}} \leftarrow \operatorname{argmax}_{\delta} [P(\{x \in B_{\delta}(v) : \hat{\mathbb{Y}}_{\text{COPAL}}[x] = \hat{\mathbb{Y}}_{\text{COPAL}}[v]\}) > \tau']$
 $\Delta \leftarrow \Delta + [\delta_{\text{opt}}]$
end for
return Δ

6.3.1 Empirical Results

We evaluate various active learning (AL) methods by training a ResNet-18 model from scratch for each strategy, utilizing labels from selected queries. In each AL iteration, a new model is trained from scratch using the updated labeled set \mathbb{L} . This process repeats across multiple AL iterations. Figure 6.1 illustrates the difference in accuracy achieved by each AL method compared to training with a random query set of the same size (baseline with no active learning) across AL iterations.

In this variation developed during our research, we did not employ the competence score $S_{\mathcal{L}}$ to weight between the objective functions $\mathcal{O}_{\text{low}}(x)$ and $\mathcal{O}_{\text{high}}(x)$ as defined in Equation 3.1. Instead, we directly used probability coverage as the weighting factor.

Here, *DCEM* refers to *DCEM* using Entropy as $\mathcal{O}_{\text{high}}(x)$ instead of Margin.

DCEM_KM_ON refers to *DCEM* incorporating the described variation of COPAL-

Kmeans.

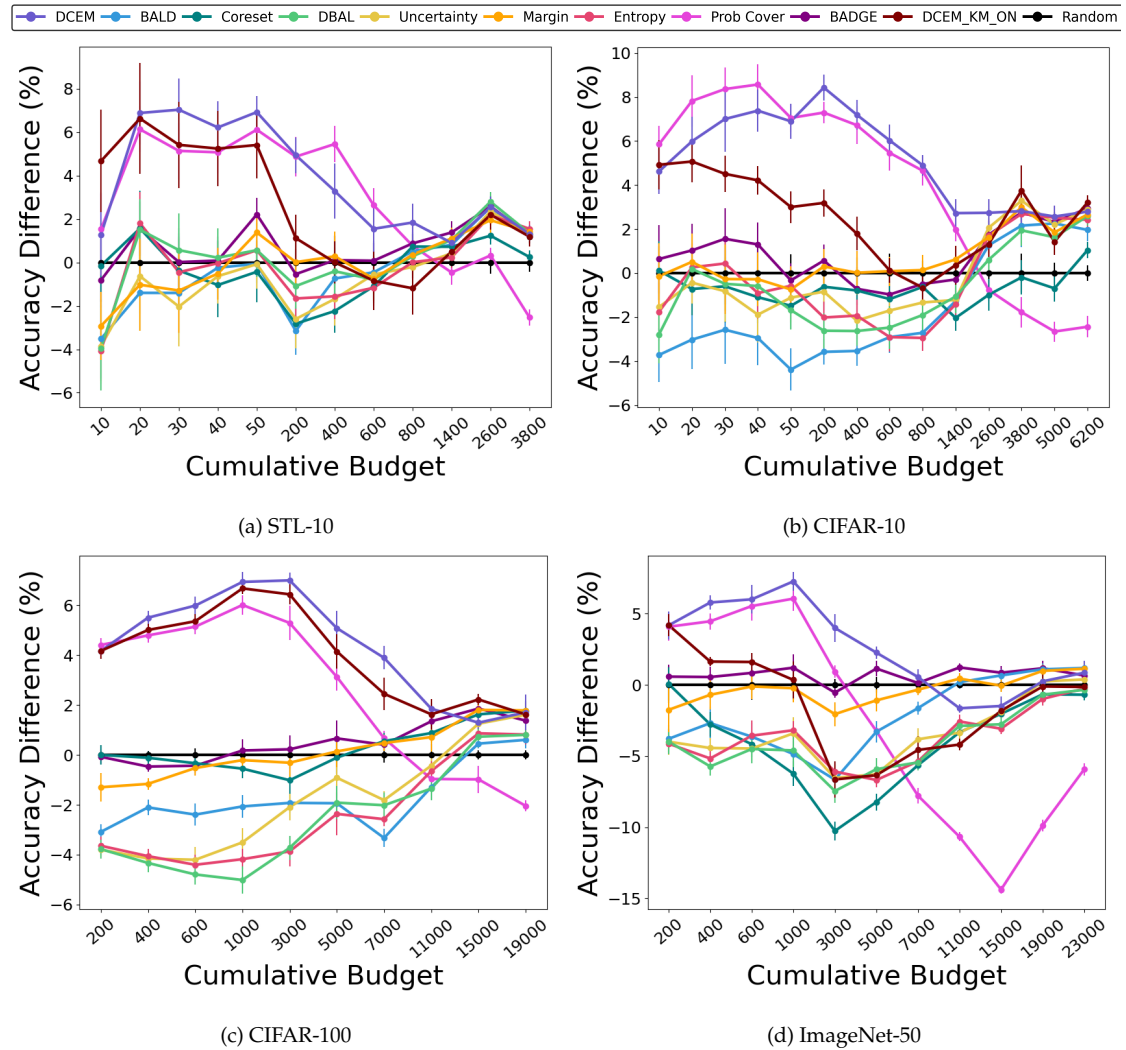


Figure 6.1: Mean accuracy difference between several AL algorithms and random queries. Positive mean difference indicates that the corresponding AL method is beneficial. The errors bars correspond to standard error, based on 5-10 repetitions (3 for the ImageNet subsets). Here you can see that the COPAL-Kmeans variation didn't improve the performance.

6.3.2 Discussions

Despite the variation not yielding improved results, this outcome may be attributed to not fully utilizing the final DCoM algorithm. Further investigation into the potential positive impact of integrating the k-means algorithm into DCoM would be of interest.

7 Summary And Conclusions

We investigate the challenge of active learning (AL) in a multi-budget setting. Our approach involves establishing a more precise constraint on the upper limit of the generalization error of a 1-nearest neighbor (1-NN) classifier. This constraint takes into account both probability coverage and coverage purity. Under certain assumptions about the data distribution, which are likely to hold in self-supervised embedding spaces, we demonstrate that optimizing both probability coverage and coverage purity minimizes the new bound and leads to the error reduction. We introduce an AL strategy dubbed *DCoM*, which approximates the optimal solution within constrained budget scenarios. Furthermore, we dynamically adjust its sampling strategy during the training process using a learner’s competence score. We validate our method empirically in both supervised and semi-supervised frameworks across various datasets. The results illustrate that *DCoM* significantly outperforms alternative methods across all budgetary constraints.

Future directions. In our future endeavors, we plan to explore:

(i) Potential avenues for enhancing the selection of a and k by leveraging existing labels or inferring a score for them through the topology of the initial self-supervised feature graph.

(ii) Soft-coverage methodologies, where the notion of coverage is probabilistic rather than binary, such as utilizing Gaussian measures.

(iii) Developing new variation of constrained K-Means that enhance cluster purity in a more efficient manner, minimizing the need for hyperparameters.

Bibliography

- [1] Angluin, D. (1988). Queries and concept learning. *Machine Learning*, 2(4):319–342.
- [2] Angluin, D. (2001). Queries revisited. In *Proceedings of the 12th International Conference on Algorithmic Learning Theory, ALT '01*, page 12–31, Berlin, Heidelberg. Springer-Verlag.
- [3] 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, SODA '07*, page 1027–1035, USA. Society for Industrial and Applied Mathematics.
- [4] Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., and Agarwal, A. (2019). Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*.
- [5] Baum, E. and Lang, K. (1992). Query learning can work poorly when a human oracle is used. In *Proceedings of the IEEE International Joint Conference on Neural Networks*. IEEE.
- [6] Chen, L., Bai, Y., Huang, S., Lu, Y., Wen, B., Yuille, A., and Zhou, Z. (2024). Making your first choice: to address cold start problem in medical active learning. In *Medical Imaging with Deep Learning*, pages 496–525. PMLR.

- [7] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.
- [8] Cho, S. J., Kim, G., Lee, J., Shin, J., and Yoo, C. D. (2023). Querying easily flip-flopped samples for deep active learning. In *The Twelfth International Conference on Learning Representations*.
- [9] Chowdhury, S., Hamerly, G., and McGarrity, M. (2024). Active learning strategy using contrastive learning and k-means for aquatic invasive species recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*, pages 848–858.
- [10] Coates, A., Ng, A., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings.
- [11] Cohn, D., Atlas, L., and Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, 15(2):201–221.
- [12] Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1996). Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145.
- [13] da Costa, V. G. T., Fini, E., Nabi, M., Sebe, N., and Ricci, E. (2022). solo-learn: A library of self-supervised methods for visual representation learning. *Journal of Machine Learning Research*, 23(56):1–6.
- [14] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

- [15] El-Hasnony, I. M., Elzeki, O. M., Alshehri, A., and Salem, H. (2022). Multi-label active learning-based machine learning model for heart disease prediction. *Sensors*, 22(3).
- [16] Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR.
- [17] Gissin, D. and Shalev-Shwartz, S. (2019). Discriminative active learning. *arXiv preprint arXiv:1907.06347*.
- [18] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent—a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284.
- [19] Hacoheh, G., Dekel, A., and Weinshall, D. (2022). Active learning on a budget: Opposite strategies suit high and low budgets. In *International Conference on Machine Learning*. PMLR.
- [20] Hacoheh, G. and Weinshall, D. (2024). How to select which active learning strategy is best suited for your specific problem and budget. *Advances in Neural Information Processing Systems*, 36.
- [21] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.
- [22] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern*

- Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.
- [23] Hounsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. (2011). Bayesian active learning for classification and preference learning. *stat*, 1050:24.
- [24] Hu, R., Mac Namee, B., and Delany, S. J. (2010). Off to a good start: Using clustering to select the initial training set in active learning. In *Twenty-Third International FLAIRS Conference*.
- [25] Huang, P., Yao, P., Hao, Z., Peng, H., and Guo, L. (2021). Improved constrained k-means algorithm for clustering with domain knowledge. *Mathematics*, 9(19).
- [26] Kaseb, A. and Farouk, M. (2023). Active learning for arabic sentiment analysis. *Alexandria Engineering Journal*, 77:177–187.
- [27] King, R., Whelan, K., Jones, F., Reiser, P., Bryant, C., Muggleton, S., Kell, D., and Oliver, S. (2004). Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252.
- [28] Kirsch, A., Van Amersfoort, J., and Gal, Y. (2019). Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32:7026–7037.
- [29] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. *Online*.
- [30] Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12.

- [31] Munjal, P., Hayat, N., Hayat, M., Sourati, J., and Khan, S. (2022). Towards robust and reproducible active learning using neural networks. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 223–232. IEEE.
- [32] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A. Y., et al. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 7. Granada, Spain.
- [33] Oquab, M., Darcet, T., Moutakanni, T., Vo, H. V., Szafraniec, M., Khalidov, V., Fernandez, P., HAZIZA, D., Massa, F., El-Nouby, A., et al. (2023). Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*.
- [34] Ranganathan, H., Venkateswara, H., Chakraborty, S., and Panchanathan, S. (2017). Deep active learning for image classification. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3934–3938.
- [35] Sener, O. and Savarese, S. (2018). Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*.
- [36] Settles, B. (2009). Active learning literature survey.
- [37] Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423.
- [38] Siddhant, A. and Lipton, Z. C. (2018). Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2904–2909.

- [39] Tharwat, A. and Schenck, W. (2023). A survey on active learning: state-of-the-art, practical challenges and research directions. *Mathematics*, 11(4):820.
- [40] Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., and Van Gool, L. (2020). Scan: Learning to classify images without labels. In *European conference on computer vision*, pages 268–285. Springer.
- [41] Wagstaff, K., Cardie, C., Rogers, S., and Schrödl, S. (2001). Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01*, page 577–584, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- [42] Wang, H., Jin, Q., Li, S., Liu, S., Wang, M., and Song, Z. (2023). A comprehensive survey on deep active learning and its applications in medical image analysis. *arXiv preprint arXiv:2310.14230*.
- [43] Wang, Y., Chen, H., Fan, Y., Sun, W., Tao, R., Hou, W., Wang, R., Yang, L., Zhou, Z., Guo, L.-Z., Qi, H., Wu, Z., Li, Y.-F., Nakamura, S., Ye, W., Savvides, M., Raj, B., Shinozaki, T., Schiele, B., Wang, J., Xie, X., and Zhang, Y. (2022). Usb: A unified semi-supervised learning benchmark for classification. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- [44] Wen, Z., Pizarro, O., and Williams, S. (2023). Ntkcpl: Active learning on top of self-supervised model by estimating true coverage. *arXiv e-prints*, pages arXiv–2306.
- [45] Woo, J. O. (2022). Active learning in bayesian neural networks with balanced entropy learning principle. In *The Eleventh International Conference on Learning Representations*.

- [46] Yang, Y., Ma, Z., Nie, F., Chang, X., and Hauptmann, A. (2015). Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113.
- [47] Yehuda, O., Dekel, A., Hacoheh, G., and Weinshall, D. (2022). Active learning through a covering lens. *Advances in Neural Information Processing Systems*, 35:22354–22367.
- [48] Yuan, D., Chang, X., Liu, Q., Yang, Y., Wang, D., Shu, M., He, Z., and Shi, G. (2023). Active learning for deep visual tracking. *IEEE Transactions on Neural Networks and Learning Systems*.
- [49] Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR.
- [50] Zhang, B., Wang, Y., Hou, W., Wu, H., Wang, J., Okumura, M., and Shinnozaki, T. (2021). Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in Neural Information Processing Systems*, 34:18408–18419.
- [51] Zhao, G., Dougherty, E., Yoon, B.-J., Alexander, F., and Qian, X. (2021). Uncertainty-aware active learning for optimal bayesian classifier. In *International Conference on Learning Representations*.