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

Active Learning Through a Covering Lens

Ofer Yehuda

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

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

December 2022

Abstract

Deep active learning aims to reduce the annotation cost for the training of deep models, which is notoriously data-hungry. Until recently, deep active learning methods were ineffectual in the *low-budget* regime, where only a small number of examples are annotated. The situation has been alleviated by recent advances in representation and self-supervised learning, which impart the geometry of the data representation with rich information about the points. Taking advantage of this progress, we study the problem of subset selection for annotation through a "covering" lens, proposing *ProbCover* – a new active learning algorithm for the low budget regime, which seeks to maximize *Probability Coverage*. We then describe a dual way to view the proposed formulation, from which one can derive strategies suitable for the high budget regime of active learning, related to existing methods like *Coreset*. We conclude with extensive experiments, evaluating *ProbCover* in the low-budget regime. We show that our principled active learning strategy improves the state-of-the-art in the low-budget regime in several image recognition benchmarks. This method is especially beneficial in the semi-supervised setting, allowing state-of-the-art semi-supervised methods to match the performance of fully supervised methods, while using much fewer labels.

Acknowledgements

First and foremost, I would like to thank my advisor, Daphna. I waited for over a year before starting to look for an advisor to work with; I was nervous, I often heard of the importance of finding the right advisor and the effect it can have on the experience. Luckily, I made the right choice, as Daphna was supportive and caring, helped set me off on and guided me in my research and organized and led a team to turn the research to a full-fledged paper. Avihu Dekel and Guy Hacohen, my co-authors, deserve special thanks for their hard work, without whom our paper would never have been published, let alone accepted to the NeurIPS conference.

I must also give credit to my friend Zako, who pushed me to return to the Hebrew University and pursue a Master's degree, and to my friend and flatmate, Guy, who witnessed on many a night my glossy eyes after another long session of debugging.

Finally, I thank my family, whom I love deeply.

Contents

List of Figures

Summary of notation

Notation	Description
$\mathbb{X}$	Input domain
$\mathbb{Y}$	Target domain
P	Probability distribution over $\mathbb{X}$
f	True labeling function
$\hat{f}$	Hypothesis for labeling function
$Loss(\hat{f})$	Generalization error
X	Unlabeled dataset
L	Labeled dataset
b	annotation budget
$B_\delta(x)$	Open ball of radius δ centered around x
δ	Ball radius
$C(L, \delta)$	The covered region. Abbreviated to C
$\pi(\delta)$	The purity of δ
$[k]$	The natural numbers $\{1, \ldots, k\}$
$p_\theta(y\|x)$	The model's label prediction distribution

Table 1: Notation Table

1 Introduction

For the most part, deep learning technology critically depends on access to large amounts of annotated data. Yet annotations are costly and remain so even in the era of *Big Data*. Deep active learning (AL) aims to alleviate this problem by improving the utility of the annotated data. Specifically, given a fixed budget b of examples that can be annotated, and some deep learner, AL algorithms aim to query those b examples that will most benefit this learner.

In order to optimally choose unlabeled examples to be annotated, most deep AL strategies follow some combination of two main principles: 1) Uncertainty sampling [e.g., 26, 49, 3, 13, 14, 36, 25], in which examples that the learner is most uncertain about are picked, to maximize the added value of the new annotations. 2) Diversity Sampling [e.g., 1, 21, 15, 40, 17, 42, 37, 16, 48, 45, 47], in which examples are chosen from diverse regions of the data distribution, to represent it wholly and reduce redundancy in the annotation.

Most AL methods fail to improve over random selection when the annotation budget is very small [35, 41, 7, 32, 55, 20, 2], a phenomenon sometimes termed "cold start" [8, 50, 15, 51, 23]. When the budget contains only a few examples, they struggle to improve the model's performance, and even fail to reach the accuracy of the random baseline. Recently, it was shown that uncertainty sampling is inherently unsuited for the low-budget regime, which may explain the cold start phenomenon [18]. The low-budget scenario is relevant in many applications, especially those requiring an expert tagger whose time is expensive (e.g., a radiologist tagger for tumor detection). If we want to expand deep learning to new domains, overcoming the cold start problem is an ever-important task.

In this work, we focus on understanding the very low budget regime of AL, where the budget of b examples cannot dependably represent the annotated data distribution. To face up to this challenge, in Sections 3.1-3.2 we model the problem as *Max Probability Cover*, defined as follows: given some data distribution, and a radius δ, select the b examples that maximize the probability of the union

of balls with radius δ around each example. We further show that under a separation assumption that is realistic in semantic embedding spaces, *Max Probability Cover* is befitting the nearest-neighbor classification model, in that it minimizes an upper bound on its generalization error.

In Section 3.3 we show a connection with existing deep AL methods, like *Coreset* [37], and explain why those methods are more suitable for the high-budget regime than the low-budget regime. This phenomenon is visualized in Fig. 3.4, where we see that with only a few examples to choose, *Coreset* – an AL strategy that employs the principle of diversity sampling – chooses distant and often abnormal points, while *ProbCover* chooses representative examples.

When using the empirical data distribution, we further show that *Max Probability Cover* can be reduced to *Max Coverage* – a known classical NP-hard problem [34] (see Section 3.2). To obtain a practical AL strategy, in Chapter 4 we adapt a greedy algorithm for the selection of b examples from a fixed finite training set of unlabeled examples (the training set), which guarantees $1 - \frac{1}{e}$ approximation to the problem. We call this new method *ProbCover*.

In Chapter 5 we empirically evaluate the performance of *ProbCover* on several computer vision datasets, including CIFAR-10, CIFAR-100, Tiny-ImageNet, ImageNet and its subsets. *ProbCover* is thus shown to significantly outperform all alternative deep AL methods in the very low-budget regime. Additionally, *ProbCover* improves the performance of state-of-the-art semi-supervised methods, which were thought until recently to make AL redundant [6], allowing for the learning of computer vision tasks with very few annotated examples.

2 Background and Related Work

2.1 Statistical Learning

2.1.1 Supervised Learning

We review the basic definitions of statistical learning [39]. We let $\mathbb{X}$ be the input domain, where the data samples reside, and $\mathbb{Y}$ be the label space, which in the classification setting is taken to be $\mathbb{Y} = [k]$. We assume there exists a distribution P over the input space $\mathbb{X}$. Usually, we are interested in some relationship between the input $\mathbb{X}$ and the labels $\mathbb{Y}$. That relationship can either be formulated as a function $f\colon \mathbb{X} \to \mathbb{Y}$, the *labeling function*, or more generally as a conditional distribution $p(y|x)$. Either way, we wish to capture that relationship, and to that end we have access to a dataset $D = \{x_i, y_i\}_{i=1}^n$, which we assume is independently identically distributed (iid) as P. Given the data, a learning algorithm will produce a hypothesis $\hat{f}\colon \mathbb{X} \to \mathbb{Y}$, which is supposed to approximate the true relationship between the inputs and the labels. Success is often measured by the misclassification error, the 0-1 loss, which in the deterministic case is

$$Loss(\hat{f}) = \mathbb{E}\left[\hat{f}(x) \neq f(x)\right]$$

As we don't assume to have access to the data distribution, the *empirical risk minimization* (ERM) principle is employed, meaning the learning algorithm seeks to minimize the *empirical loss* on the training set

$$Loss_{emp}(\hat{f}) = \frac{1}{n}\sum_i \mathbb{1}_{\hat{f}(x_i)\neq y_i}$$

Often, the output of the learning algorithm is not a predictor but a parametrized probability distribution over the possible labels $p_\theta(y|x)$. The predictor is then set to output the label with the highest probability

$$\hat{f}(x) = \text{argmax}_y \, p_\theta(y|x)$$

An example of a model which produces such such distribution is *logistic regression*.

2.1.2 Semi-Supervised learning

In the *semi-supervised learning* scenario, the learner has access to a large unlabeled pool of data X, and a small pool of labeled data L. The aim is then to leverage the unlabeled pool X to improve upon the performance possible when using only labeled data. Semi-supervised learning is one learning formulation for the scenario where labeled data is scarce but unlabeled data is abundant.

Deep semi-supervised learning is a subfield with methods designed specifically for use with neural network models. In our benchmarks we use a semi-supervised algorithm called *FlexMatch* [52], which is an elaboration on a semi-supervised algorithm called *FixMatch*[43].

In *FixMatch*, the idea is to generate pseudolabels for part of the unlabeled data, and then train a classifier $p_\theta(y|x)$ to predict them as well as the labeled data. In a sense, it tries to increase the size of the labeled pool by turning unlabeled data into labeled data.

To generate a pseudolabel for a given unlabeled sample x, the algorithm creates two augmentations of the sample: a weak augmentation $\omega_w(x)$ and a strong augmentation $\omega_s(x)$ (the weak/strong terminology refers to how different the resulting augmentation is from the original sample). The classifier then predicts a label for the weak augmentation, and this prediction becomes the pseudolabel.

$$\tilde{y} = \text{argmax}_y \, p_\theta(y|\omega_w(x))$$

The authors observed that generating pseudolabels for the entire unlabeled pool was detrimental to the performance, so they added a filtering mechanism to improve the quality of the pseudolabels. A sample only gets a pseudolabel if the classifier is confident about its prediction from the weak augmentation. Specifically, the pseudolabel is only generated if the prediction probability from the

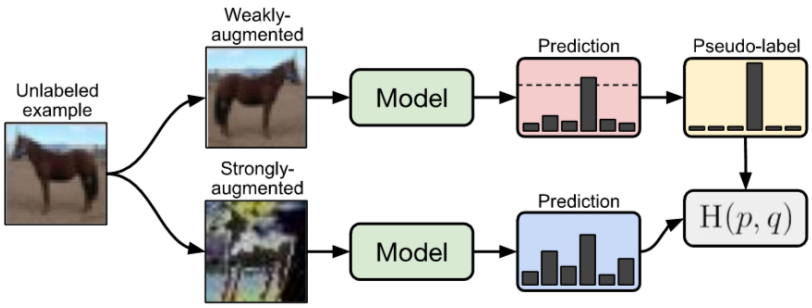


Figure 2.1: Illustration of *FixMatch* pseudolabel creation and auxiliary loss.

weak augmentation is greater than a fixed threshold T (which is a hyperparameter determining the required confidence level):

$$p_\theta(\tilde{y}|\omega_w(x)) \geq T$$

The unlabeled training loss is then the cross entropy

$$\sum_{x \in U} \mathbb{1}_{p_\theta(\tilde{y}|\omega_w(x)) \geq T} H(\mathbb{1}_{\tilde{y}}, p_\theta(\cdot|\omega_s(x))$$

In the *FlexMatch* paper, the authors improved upon *FixMatch* by adding an adaptive class specific threshold. This alleviates some of the problem *FixMatch* suffers from, like imbalanced distribution of pseudolabels, and unstable training.

2.1.3 Unsupervised and Self-Supervised learning

In the *unsupervised learning* framework, the only data available is unlabeled data X. Without the guidance of labels, it is often used to find structure or patterns in the data. Unsupervised learning algorithms are often applied to problems such as dimensionality reduction, clustering and anomaly detection.

Self-supervised learning (SSL) is a form of unsupervised learning. It generates and trains on auxiliary tasks from unlabeled data. The network can be pretrained on these tasks in a supervised manner, and can later be used as a learned representation, or further fine-tuned on a smaller amount of labeled data. SSL techniques have been shown to be effective in learning useful representations for text and image data. For representation learning, one of the final layers of the trained model is taken to be the input's representation, usually after some sort of normalization.

In our experiments we make use of two SSL techniques: *SimCLR*[9] and *DINO*[5]. SimCLR defines a contrastive auxiliary task, where the goal is to maximize similarity between different views of the same data point, while minimizing similarity to views from other data points. The views are generated using augmentations: positive pairs are two augmentations from the same sample, and negative pairs are augmentations from other samples. During training, the model predicts which of the samples presented is the positive pair. The contrastive loss encourages the representation of positive pairs to be similar while encouraging negative pairs representation to be dissimilar.

DINO is another method inspired by the concept of *knowledge distillation*. It involves training a student network to mimic the output of a teacher network. To generate the training data, a set of crop augmentations is generated from an image. Two of the augmentations are global, comprising most of the image, and are passed through the teacher. The remaining crops are local and passed through the student. A cross-entropy loss is employed between the student and teacher's output. This encourages the student network to learn features discriminating local and global properties of images.

In knowledge distillation, the teacher network is a pretrained model, so the student's task of predicting the teacher's outputs can be seen as "distilling" the teacher. In the SSL scenario, the teacher has to be bootstrapped during training. In DINO the teacher's parameters are periodically updated from an exponentially moving average of the student's parameters. To prevent the phenomena of "model collapse", where all the outputs are trivially mapped to the same vector, two measures are taken: The teacher's outputs are mean-centered over the batch, and the outputs are "sharpened" using low-temperature softmax. This measures help ensure the stability and success of the training.

2.1.4 Active Learning

In the previous sections, the learning framework was static, with the unlabeled and labeled data a fixed part of the learning problem. *Active learning* adds an interactive element to the learning process, by letting the learner query for additional labels during the learning process. This is similar to how a student can ask a teacher a question to improve their understanding.This way, the learner can adaptively select the most informative samples, instead of relying on a fixed set of labeled samples, as in unsupervised learning, with the hope that this will result in accelerated learning.

The training process in active learning is iterative. First, the learner is supplied

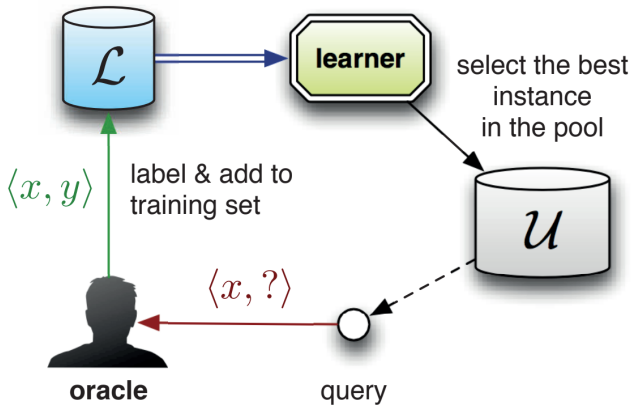


Figure 2.2: Illustration of the active learning loop [38].

with an unlabeled pool and a smaller pool of labeled examples (which may be empty). It then trains on the available data and uses the resulting predictor to query additional samples from the unlabeled pool. The process is repeated for a number of rounds determined by the *budget* b available. In the end, the learner produces a predictor trained on the final unlabeled and labeled pools. Figure 2.2 visualizes the process.

A distinction between "high" and "low" budget scenarios has been made in the context of deep learning. Informally, the budget is low when it is in the similar order of magnitude to the number of classes. As deep learning requires large amounts of data, Hacohen et al. [18] observed that most active learning strategies failed to improve deep models over the random baseline when the budget was low. They provided an analysis of this distinction and showed that the different scenarios require different approaches. While uncertainty based approaches work well in the high budget, geometry-based approaches, such as clustering, are more effective in the low budget.

Active learning has been a field of study since the 1990s and includes a wide variety of works, ranging from theoretical analysis under the statistical learning framework to more practical, heuristic-based methods. We next describe the prominent active learning methods against which we compare in our experiments.

Uncertainty sampling methods

Uncertainty based methods query samples that the model is least certain about, based on various criteria. These include *least confidence*, *max-margin* and *entropy*. They each measure prediction uncertainty in a different way. For a sample x with predicted label $\hat{y} = \text{argmax}_y\, p_\theta(y|x)$, the scores are:

- Least confidence - $1 - p_\theta(\hat{y}|x)$
- Max-margin - $p_\theta(\hat{y}|x) - \max_{y \neq \hat{y}} p_\theta(y|x)$
- Entropy - $H(p_\theta(\cdot|x))$

BADGE [1] is a method designed for deep learning models. It is an uncertainty sampling based method, with some consideration of diversity. The idea behind the method is to use the weights' gradient of the loss for each sample as the source of information for uncertainty. They compute a pseudolabel $\hat{y}(x)$ for each sample x and calculate the gradient $g_\theta(x)$ of the last layer with respect to the loss on the pseudolabel $-\log p_\theta(\hat{y}|x)$. This gradient is then taken as an embedding vector. Furthermore, this embedding can be thought of as an uncertainty embedding: uncertain predictions results in larger gradient norms. The *k-means++* algorithm is then used to select samples from the gradient embedding space. This algorithm is an initialization scheme for k-means, so it naturally favors diverse and large norm embeddings.

The uncertainty approach can also be used in the Bayesian framework. *DBAL* [14] (Deep Bayesian Active Learning) is a method for doing Bayesian uncertainty estimation on CNNs. In the paper the authors show that *dropout* layers are equivalent to doing Monte-Carlo approximations of the posterior uncertainty and other bayesian measures. In our experiments DBAL refers to the use of the method for approximating the Bayesian entropy measure

$$H(y|x, L)$$

We also use the method to approximate and compare against *BALD* [22] (Bayesian Active Learning by Disagreement), another aquisition function defined as

$$I[y, \theta|x, L] = H(y|x, L) - \mathbb{E}_\theta[H(y|x, \theta)]$$

This quantity is maximized when $H(y|x, L)$ is high, so the prediction of y from x given the labeled data L is uncertain, and $\mathbb{E}_\theta[H(y|x, \theta)]$ is low, so individual

parameter produce confident predictions of y. Thus it maximizes disagreement between different parameters/models.

Diversity sampling methods and batch queries

When it comes to deep learning, the single point query setting is unrealistic, as training is performed by gradient descent on minibatches of data, and adding one sample has little effect on performance, while the training is costly. Therefore, some strategies choose a batch of points to query at once. Querying in batches introduces the problem of redundancy, where many points are informative on their own but overlapping in information. As an example, consider the case where uncertainty sampling is used. All the queried points in the batch might have a high uncertainty, but if they are extremely similar then the added value of the batch is diminished. Hence, there must be some consideration for the *diversity* of the queried batch.

Coreset [37] is a diversity-based method: it aim to select a labeled set L that minimize the *maximum facility location distance*

$$\max_{u \in X} \min_{l \in L} d(u, l)$$

which intuitively causes selected samples to come from diverse regions of the space. Since it is a NP-hard goal the actual method uses a greedy approximation. Since distances in high-dimensional input spaces are are often non-informative, the method uses as an embedding the activations of the last few layers of the trained model.

TypiClust [18] is a method intended for the low-budget active learning scenario. It uses learned self-supervised representation, and clusters the embeddings into b different clusters, choosing from each cluster the densest example.

W-Dist [30] is another low-budget active learning method, that aims to choose a labeled set L that minimizes the *Wasserstein* distance to the unlabeled set. As the objective is a MIP optimization problem, the authors derive and use an iterative approximation scheme.

2.2 Maximum Coverage

In this work we refer to *Max Coverage*, a known NP-hard problem. We recapitulate its definition as follows:

Definition 2.2.1 (*Max Coverage*)**.** Let $b \in \mathbb{N}$ denote an integer, U denote a set of elements, and let $S = \{S_1, \ldots, S_m\}$ denote a collection of subsets of U. In the problem of *Max Coverage* we wish to find b subsets in S with union of maximum cardinality

$$\underset{S' \subseteq S; |S'|=b}{\operatorname{argmax}} \left| \bigcup_{S_i \in S'} S_i \right|$$

3 Theory

There is already a large body of work analysing active learning algorithms through the lens of statistical learning, but similarly to the case with deep learning in general, there is still a large gap between theory and practice. In light of that, we choose in this work a different, geometrical and distribution-based approach to motivate our suggested active learning method. We also incorporate the existence of semantic embedding spaces through assumptions on the "niceness" of the label distribution.

Nearest-Neighbor models are often used to compare the quality of supervised and self-supervised representations, based on the intuition that a good representation allows classification with simple models. We extend this notion further, and consider the accuracy of 1-NN models as a proxy for the labeled examples' informativeness. With this assumption we can mathematically analyse the effect of different sample sets on performance, and suggest a theoretically-motivated AL method.

3.1 Bounding the Generalization Error

We restate some of the notation, and lay a few definitions. Most important is the assumption of δ-purity, which states that most of the time, points that are less than δ apart have the same label. We then prove a lemma, showing that given a labeled set L and the coverage it achieves, and given the δ-purity assumption, the probability of a point being inside this cover and still being falsely labeled is small. From this, we finally derive a bound on the generalization error, which is stated in Thm. 1.

Notations Let $\mathbb{X}$ denote the input domain whose underlying probability function is denoted P, and let $\mathbb{Y} = [k]$ denote the target domain. Assume that a true labeling function $f : \mathbb{X} \to \mathbb{Y}$ exists. The *Generalization Error* of a hypothesis $\hat{f} : \mathbb{X} \to \mathbb{Y}$ is defined as $Loss(\hat{f}) = \mathbb{E}\left[\hat{f}(x) \neq f(x)\right]$. Let $X = \{x_i\}_{i=1}^{m}$ denote an unlabeled set of points, and $b \leq m$ the annotation budget. Let $L \subseteq X$ denote the labeled set, where $|L| = b$. Let $B_\delta(x) = \{x' : \|x' - x\|_2 \leq \delta\}$ denote a ball centered at x of radius δ. Let $C \equiv C(L, \delta) = \bigcup_{x \in L} B_\delta(x)$ denote the region covered by δ-balls centered at the labeled examples in L. We call $C(L, \delta)$ the *covered region* and $P(C)$ the *coverage*.

Definition 3.1.1. We say that a ball $B_\delta(x)$ is *pure* if $\forall x' \in B_\delta(x) : f(x') = f(x)$.

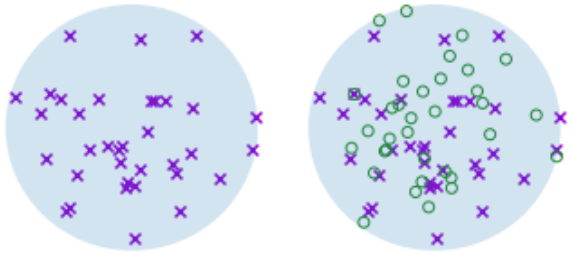


Figure 3.1: Pure and impure balls. On the left is a *pure* ball, containing labels from only one class. On the right is an impure ball, with labels from two classes.

Definition 3.1.2. We define the *purity* of δ as: $\pi(\delta) = P(\{x : B_\delta(x) \text{ is pure}\})$.

Remark. $\pi(\delta)$ is monotonically decreasing.

Let $\hat{f}$ denote the 1-NN classifier based on L. We split the covered region $C(L, \delta)$ into two sets:

$$C_{true} = \{x \in C : \hat{f}(x) = f(x)\},\ C_{false} = C \setminus C_{true}.$$

Lemma 1. $C_{false} \subseteq \{x : B_\delta(x) \text{ is not pure}\}$.

Proof. Let $x \in C_{false}$. Let $c \in L$ denote the nearest neighbor to x. Then they have the same predicted label, $\hat{f}(x) = \hat{f}(c)$. Furthermore, since c is labeled $f(c) =$

$\hat{f}(c)$. $x \in C_{false}$, meaning it is wrongly labeled by $\hat{f}$,

$$f(c) = \hat{f}(c) = \hat{f}(x) \neq f(x).$$

Finally, since $x \in C_{false} \subseteq C$ is in the coverage, $d(x, c) < \delta$, which means that $c \in B_\delta(x)$ with a different label and so $B_\delta(x)$ is not pure. □

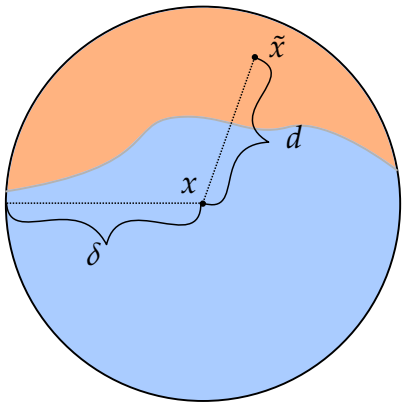


Figure 3.2: Proof illustration for lemma 1. $\tilde{x} \in C_{false}$ while x is its closest labeled neighbor. Both have different labels (indicated by the colors), and their distance d is smaller than δ, so the ball $B_\delta(\tilde{x})$ is not pure.

Corollary 1.

$$P(C_{false}) \leq P(\{x : B_\delta(x) \text{ is not pure}\}) = 1 - \pi(\delta).$$

Theorem 1. The generalization error of the 1-NN classifier $\hat{f}$ is bounded by

$$Loss(\hat{f}) \leq (1 - P(C(L, \delta))) + (1 - \pi(\delta)). \tag{3.1}$$

Proof.

$$\begin{aligned}
Loss(\hat{f}) &= \mathbb{E}[\mathbb{1}_{f(x)\neq\hat{f}(x)}] = \mathbb{E}[\mathbb{1}_{f(x)\neq\hat{f}(x)}\mathbb{1}_{x\notin C}] + \mathbb{E}[\mathbb{1}_{f(x)\neq\hat{f}(x)}\mathbb{1}_{x\in C}] \\
&\leq P(x \notin C) + \mathbb{E}[\mathbb{1}_{f\neq\hat{f}}\mathbb{1}_{x\in C_{true}}] + \mathbb{E}[\mathbb{1}_{f(x)\neq\hat{f}(x)}\mathbb{1}_{x\in C_{false}}] \\
&\leq P(x \notin C) + 0 + P(x \in C_{false}) \\
&\leq (1 - P(C(L,\delta))) + (1 - \pi(\delta)).
\end{aligned}$$

□

Note that (1) gives us a different bound for different δ values, and also depends on the labeled set L. This bound introduces a trade-off: as δ increases, the *coverage* increases, but the *purity* decreases. Therefore, we wish to find the pair δ, L that gives the tightest bound.

We can interpret (3.1) in the context of two boundary conditions of AL: high-budget and low-budget. In the high-budget regime, achieving full coverage $P(C) = 1$ is easy as we have many points, and the remaining challenge is to reduce $1 - \pi(\delta)$. Since $\pi(\delta)$ is monotonically decreasing, we can seek to minimize δ subject to the constraint $P(C) = 1$. This is similar to *Coreset* [37]. In the low-budget regime, full coverage entails very low purity, which (if sufficiently low) makes the bound trivially 1. Thus, instead of insisting on full coverage, we can fix a δ that yields "large enough" purity $\pi(\delta) > 0$, and then seek a labeled set L that maximizes the coverage $P(C)$. We call this problem *Max Probability Cover*.

3.2 Max Probability Cover

Definition 3.2.1 (*Max Probability Cover*)**.** Fix $\delta > 0$, and obtain a subset $L \subset X$, $|L| = b$, that maximizes the probability of the covered space

$$\underset{L\subseteq X;|L|=b}{\operatorname{argmax}} P(\bigcup_{x\in L} B_\delta(x)) \tag{3.2}$$

An optimal solution to (3.2) would minimize the bound in (3.1), when δ is fixed (thus fixing $1 - \pi(\delta)$).

Unfortunately, when moving to practical settings, there are two obstacles. The

first is complexity:

Theorem 2. *Max Probability Cover* is NP-hard.

The second problem is that the data distribution is hardly ever known apriori, and even if it were, the probability computation would likely be intractable and hard to approximate. Instead, we may use the *empirical distribution* $\tilde{P}(A) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{x_i \in A}$ as an approximation, to get the following result:

Proposition. When P is taken to be the empirical distribution $\tilde{P}$, the *Max Probability Cover* objective function is equivalent to the *Max Coverage* 2.2.1 objective, with $\{B_\delta(x_i) \cap X\}_{i=1}^{n}$ as the collection of subsets.

Proof. Given a labeled set $L = \{x_i\}_{i=1}^{b}$, we show equality of objectives up to constant $\frac{1}{|X|} = \frac{1}{b}$

$$\begin{aligned} P\left(\bigcup_{i=1}^{b} B_\delta(x_i)\right) &= P\left(\{y \in \mathbb{R}^d \mid \exists i \quad \|x_i - y\| < \delta\}\right) \\ &= \frac{1}{|X|}\left|\{x \in X \mid \exists i \quad \|x_i - x\| < \delta\}\right| \\ &= \frac{1}{b}\left|\bigcup_{i=1}^{b}(B_\delta(x_i) \cap X)\right| \end{aligned}$$

□

3.2.1 *Max Probability Cover* is NP-Hard

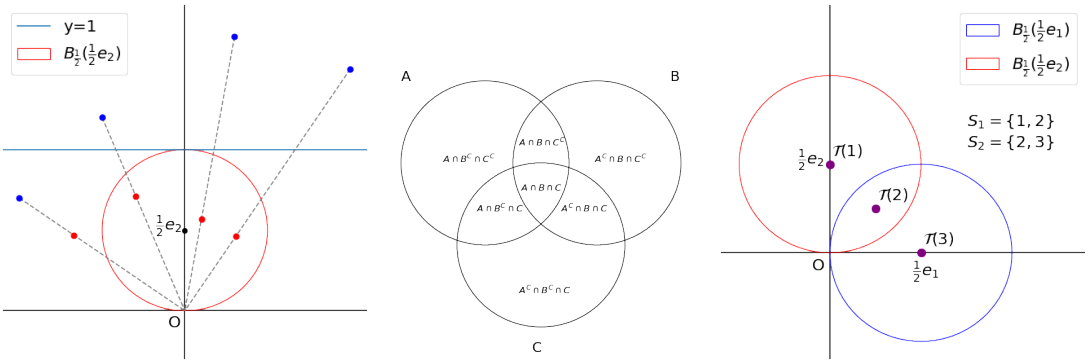
We first describe a constructive procedure to generate a collection of balls in $\mathbb{R}^m$ with a property we call *exhaustive intersection* (Def. 3.2.3). We then use this collection in order to construct a reduction from *Max Coverage* to *Max Probability Cover*.

Exhaustive Intersection: Constructive Procedure

Let $\{e_i\}_{i=1}^{m}$ denote the natural basis of $\mathbb{R}^m$. Let $B_r(p)$ denote the open ball of radius r around p, and Let $B_r[p]$ similarly denote the closed ball.

Definition 3.2.2 (Inversion mapping)**.** We define the *inversion* mapping $\iota : \mathbb{R}^m \setminus \{0\} \to \mathbb{R}^m \setminus \{0\}$ as

$$\iota(p) = \frac{p}{\|p\|_2^2}$$



(a) (b) (c)

Figure 3.3: (a) Visualization of Lemma 2. The blue points above the plane $y = 1$ are mapped by $\iota(\cdot)$ to the red points inside $B_{\frac{1}{2}}(\frac{1}{2}e_2)$. Points below $y = 1$ are mapped outside the ball. (b) The exhaustive intersection in $\mathbb{R}^2$. (c) Visualization of the induced distribution for $m = 2$. Points $1, 3$ are mapped to disjoint parts of the two balls, while point 2 is mapped to their intersection $B_{\frac{1}{2}}(\frac{1}{2}e_1) \cap B_{\frac{1}{2}}(\frac{1}{2}e_2)$. Each point is then assigned a Dirac measure and the distribution is the normalized sum, as a result of which we get that $P(B_{\frac{1}{2}}(\frac{1}{2}e_1)) = P(B_{\frac{1}{2}}(\frac{1}{2}e_2)) = \frac{2}{3}$.

Lemma 2. Let $HS_i = \{x \subseteq \mathbb{R}^m \mid x \cdot e_i > 1\}$ denote a halfspace. Then $\iota(HS_i) \subseteq B_{\frac{1}{2}}(\frac{1}{2}e_i)$.

Proof. Let $x \in HS_i$. Membership in the two sets is defined by satisfying the equations $x \cdot e_i > 1$ and $d(\iota(x), \frac{1}{2}e_i) < \frac{1}{2}$. We will show that they are equivalent (see visualization in Fig. 3.3a). We use the polarization identity $a \cdot b = \frac{1}{2}(\|a\|^2 + \|b\|^2 - d(a, b)^2)$

$$
\begin{aligned}
1 &< x \cdot e_i \\
\frac{1}{2\|x\|^2} &< \left(\frac{x}{\|x\|^2}\right) \cdot \left(\frac{1}{2}e_i\right) \\
\frac{1}{2\|x\|^2} &< \frac{1}{2}\left(\frac{1}{\|x\|^2} + \frac{1}{4} - d\Big(\frac{x}{\|x\|^2}, \frac{1}{2}e_i\Big)^2\right) \\
d\left(\frac{x}{\|x\|^2}, \frac{1}{2}e_i\right)^2 &< \frac{1}{4} \\
d\left(\iota(x), \frac{1}{2}e_i\right) &< \frac{1}{2}
\end{aligned}
$$

□

Corollary 2. $x \neq 0,\ x \cdot e_i < 1 \iff \iota(x) \notin B_{\frac{1}{2}}[\frac{1}{2}e_i]$.

Definition 3.2.3 (Exhaustive intersection)**.** A collection of sets $(A_1, \ldots, A_m)$ is said to have the *exhaustive intersection* property if for any subset of indices $I \subset [m]$ there exists a point x_I with

1. $x_I \in \bigcap_{i \in I} A_i$
2. $x_I \notin \bigcup_{j \in [m] \setminus I} A_j$

Put differently, $x_I \in A_i \iff i \in I$ (see the Venn diagram example in Fig. 3.3b).

Lemma 3. The collection of balls $\{B_{\frac{1}{2}}(\frac{1}{2}e_i)\}_{i=1}^m$ in $\mathbb{R}^m$ satisfies the *exhaustive intersection* property.

Proof. Let $I \subseteq [m]$ denote a subset of indices and define $\tilde{x}_I = \sum_{j \in I} 2e_j$, $x_I = \iota(\tilde{x}_I)$. From Lemma 2, for any $j \in I$, $\tilde{x}_I \cdot e_j = 2 \Rightarrow x_I \in B_{\frac{1}{2}}(\frac{1}{2}e_j)$, which proves the first part of Def. 3.2.3. From Cor. 2, any $j \notin I$, $\tilde{x}_I \cdot e_j = 0 \Rightarrow x_I \notin B_{\frac{1}{2}}(\frac{1}{2}e_j)$, which proves the second part. □

Reduction from *Max Coverage* to *Max Probability Cover*

Before we start, we note that in order for *Max Probability Cover* to be computable and not be trivially polynomial, we must assume that the encoding of the input distribution to *Max Probability Cover* is of polynomial size in the number of points.

Theorem 2. *Max Probability Cover* is NP-hard.

Proof. We construct a polynomial-time reduction from any *Max Coverage* problem $\mathbb{P}$ to another problem $\tilde{\mathbb{P}}$, which is an instance of *Max Probability Cover*. Let $In = (S_1, \ldots, S_m, b)$ be the input to $\mathbb{P}$, and let $U = \bigcup_{i=1}^{m} S_i$.

We define a mapping from the input of $\mathbb{P}$ to the input of $\tilde{\mathbb{P}}$: First, we fix the input space of $\tilde{\mathbb{P}}$ to $\mathbb{R}^m$, where m is the number of sets in $\mathbb{P}$, and fix the set of m points X in $\tilde{\mathbb{P}}$ to $\{\frac{1}{2}e_i\}_{i=1}^{m} \subseteq \mathbb{R}^m$. We let b remain the same, and fix the radius $\delta = \frac{1}{2}$. We abbreviate $B_i = B_{\frac{1}{2}}(\frac{1}{2}e_i)$. Next, we define a mapping $\mathcal{T} : U \to \mathbb{R}^m$ as $\mathcal{T}(u) = \iota(\sum_{i=1}^{m} 2 \cdot \mathbb{1}_{u \in S_i} e_i)$. From the proof of Lemma 3 we conclude that

$$\mathcal{T}(u) \in B_i \iff u \in S_i$$

Finally, we define the probability distribution P as $P(A) = \frac{1}{|U|} \sum_{u \in U} \mathbb{1}_{\mathcal{T}(u) \in A}$ (see visualization in Fig. 3.3c).

Lemma 4. *P is a valid probability distribution.*

Proof. P is a finite sum of Dirac measures $\mathbb{1}_{\mathcal{T}(u) \in A}$, and as such it is a measure itself. Hence we only need to show that it is normalized, ie $P(\mathbb{R}^m) = 1$:

$$P(\mathbb{R}^m) = \frac{1}{|U|} \sum_{u \in U} \mathbb{1}_{\mathcal{T}(u) \in \mathbb{R}^m} = \frac{1}{|U|} \sum_{u \in U} 1 = 1$$

□

In summary, the input of $\tilde{\mathbb{P}}$ is the following:

- Dataset: $X(In) = \{\frac{1}{2}e_i\}_{i=1}^{m} \subseteq \mathbb{R}^m$.
- Budget: $b(In) = b$.
- Ball radius: $\delta(In) = \frac{1}{2}$.
- Distribution: $(P(In))(A) = \frac{1}{|U|}\sum_{u\in U} \mathbb{1}_{\mathcal{T}(u)\in A}$

Before we continue, we require another short lemma:

Lemma 5. For all $u \in U$, $I \subseteq [m]$, $\mathbb{1}_{\mathcal{T}(u)\in\bigcup_{i\in I} B_i} = \mathbb{1}_{u\in\bigcup_{i\in I} S_i}$

Proof. We prove the conditions are equivalent:

$$\mathcal{T}(u) \in \bigcup_{i\in I} B_i \iff \exists i \in I \quad \mathcal{T}(u) \in B_i \iff \exists i \in I \quad u \in S_i \iff u \in \bigcup_{i\in I} S_i$$

□

To show that a reduction is valid for an optimization problem, we need to show that the objectives are equivalent. The objective in *Max Coverage* is the size of the union, whereas in *Max Probability Cover* it is the probability of the δ-ball union.

$$\begin{aligned} P(\bigcup_{i\in I} B_i) &= \frac{1}{|U|}\sum_{u\in U} \mathbb{1}_{\mathcal{T}(u)\in\bigcup_{i\in I} B_i} \\ &= \frac{1}{|U|}\sum_{u\in U} \mathbb{1}_{u\in\bigcup_{i\in I} S_i} \\ &= \frac{1}{|U|}\left|\bigcup_{i\in I} S_i\right| \end{aligned}$$

Since $|U|$ is constant the optimization is equivalent.

Finally, we show that the induced probability can be specified in polynomial space: as $|U|$ is polynomial in the size of the input, it follows that the distribution as a normalized sum of indicator functions can be specified as a table of

the embedding of size $|U| \cdot m$, which is polynomial. □

3.3 The "Duality" of *Max Probability Cover* and *Coreset*

The *Coreset* AL method by Sener et al. [37] minimizes the objective

$$\delta(L) = \max_{x \in X} \min_{c \in L} d(x, c) = \min\{\delta \in \mathbb{R}_+ \colon X \subseteq \bigcup_{c \in L} B_\delta(c)\}$$

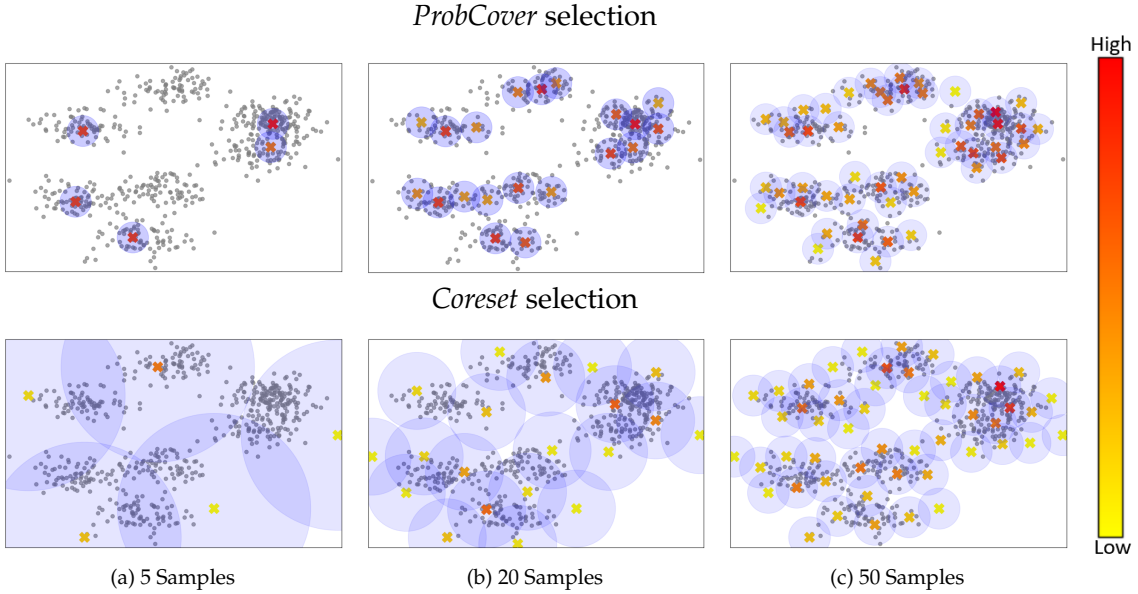
We can rewrite the above in the language of distributions as

$$\delta'(L) = \min\{\delta \in \mathbb{R}_+ \colon P(\bigcup_{c \in L} B_\delta(c)) = 1\}$$

If we use the empirical distribution then $\delta(L) = \delta'(L)$. In this framework we can say that *Max Probability Cover* and *Coreset* are dual problems (in a loose sense) as follows:

1. *Max Probability Cover* minimizes the generalization error bound (3.1) when we fix δ and seek to maximize the coverage, which is suitable for the low budget regime.
2. *Coreset* minimizes the generalization error bound (3.1) when we fix the coverage to 1 and minimize δ, which is suitable for the high budget regime because only then can we fix the coverage to 1.

This duality is visualized in Fig. 3.4.



(a) 5 Samples (b) 20 Samples (c) 50 Samples

Figure 3.4: *ProbCover* selection (top) vs *Coreset* selection (bottom) of 5/20/50 samples (out of 600). Selected points are marked by **x**, which is color-coded by density (see color code bar to the right). Density is measured using Gaussian Kernel Density Estimation, and the covered area is marked in light blue. *Coreset* attempts to minimize ball size, constrained by complete coverage, while *ProbCover* attempts to maximize coverage, constrained by a fixed ball size. Note that especially in low budgets, such as 5 samples, *Coreset* only selects outliers of the distribution (yellow), while *ProbCover* selects from dense regions of the distribution (red).

4 Methods

To deliver a practical method, we first note that our approach implicitly relies on the existence of a good embedding space [4, 10, 54], where distance is correlated with semantic similarity, and where similar points are likely to bunch together in high-density regions. As is now customary [e.g., 30, 18], we use an embedding space derived by training a self-supervised task over the large unlabeled pool. In such a space similar labels often correspond to short distances, making 1-NN classification suitable, and also providing for the existence of large enough δ balls with good purity and coverage properties.

Secondly, we note that *Max Coverage* is NP-hard and cannot be solved efficiently. Instead, as its objective is submodular and monotone [27], we use the greedy approximate algorithm that achieves $\left(1 - \frac{1}{e}\right)$-approximation [27]. A better approximation is impractical, as shown in App. B.1.

Below, we describe the greedy algorithm in Section 4.1, and the estimation of ball size δ in 4.1.2.

4.1 Greedy Algorithm

The algorithm (see Alg. 1 for pseudo-code) proceeds as follows: First, construct a directed graph $G = (V, E)$, with $V = X$ being the embedding of the dataset, and $(x, x') \in E \iff x' \in B_\delta(x)$. In G, each vertex represents a specific example, and there is an edge between two vertices (x, x') if x' is covered by the δ-ball centered at x (distances are measured in the embedding space). The algorithm then performs b iterations of the following two steps: (i) pick the vertex x_{max} with the highest out-degree for annotation; (ii) remove all incoming edges to x_{max} and its neighbors.

Algorithm 1 *ProbCover*

Input: unlabeled pool X, labeled pool L, budget b, ball-size δ
Output: a set of points to query
$X \leftarrow$ Embedding of representation learning algorithm on $X \cup L$
$G = (V = X, E = \{(x, x') : x' \in B_\delta(x)\})$
for all $c \in L$ **do**
  Remove the incoming edges to covered vertices, $\{(x', x) \in E : (c, x) \in E\}$, from E
end for
Queries $\leftarrow \emptyset$
for all i=1,...,b **do**
  Add $c \in X$ with the highest out-degree in G to *Queries*
  Remove the incoming edges to covered vertices, $\{(x', x) \in E : (c, x) \in E\}$, from E
end for
return *Queries*

4.1.1 Time and Space complexity

ProbCover is run once before the training loop begins. It selects the best subset to be labeled by humans, which is then used to train the model.

For the complexity calculation below, let n denote the number of examples in the unlabeled and labeled pools $|X \cup L|$, d the dimension of the data embedding space, and b the query budget. ProbCover can be split into two steps:

Adjacency graph

Constructing the adjacency graph requires computing pairwise distances in the embedding space.

Time Complexity: $O(n^2 d)$ time. In practice, it takes roughly 10 minutes on a single NVIDIA A4000 GPU even on the largest dataset we consider – ImageNet with DINO embedding, where $n = 1281167$, $d = 384$.

Space Complexity: naively we have $O(n^2)$, which is impractical for large datasets like ImageNet. However, we only need to save edges whose distance is smaller than δ. We store the edges using a sparse matrix in coordinate list (COO) format, so the space complexity is $O(|E|)$, where E is the set of edges in the graph.

Although $O(|E|)$ is still $O(n^2)$ in the worst case, in practice, the average degree of each vertex in the graph using radius δ is a few orders of magnitude smaller than n, resulting in manageable space complexity. For example, When selecting sam-

ples from ImageNet with $\delta = 0.55$, the average degree was 24 and the algorithm total memory consumption was $12GB$.

Sample selection

We iteratively select samples from the current sparse graph, removing incoming edges to newly covered samples. We note that unlike the adjacency graph creation, the sample selection cannot be parallelized, as each selection step depends on the previous step.

Time Complexity

Breaking down the steps in the selection of a single sample:

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

All in all, the complexity is $O(|E| + n)$ for selecting a sample, and $O(n^2k)$ in the worst case. As we select more and more points, more edges are removed, making the selection of later samples faster. In practice, thanks to the vectorization of these steps it takes roughly 15 minutes to select $k = 1000$ samples from ImageNet on a single CPU, and a couple of seconds from CIFAR-10/100.

4.1.2 Estimating δ

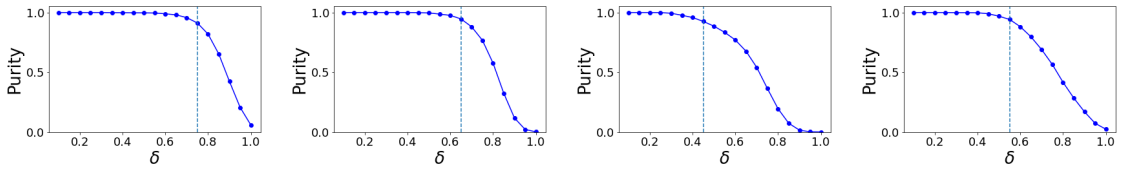
Our algorithm requires the specification of hyper-parameter δ, the ball radius, whose value depends on details of the embedding space (see App. A.1 for embeddings used). In choosing δ, we need to consider the trade-off between large coverage $P(C)$ and high purity $\pi(\delta)$. We resolve this trade-off with the following heuristic, where we pick the largest δ possible, while maintaining purity above a certain threshold $\alpha \in (0, 1)$. Specifically,

$$\delta^* = \max \{\delta : \pi(\delta) \geq \alpha\}$$

Importantly, α is more intuitive to tune, and is kept constant across different datasets (unlike δ). We still need to estimate the purity $\pi(\delta)$, which depends on the labels, from unlabeled data. To this end, we estimate purity using unsupervised representation learning and clustering. First, we cluster self-supervised features using k-means with k equal to the number of classes. For a given δ, we

compute the purity $\pi(\delta)$ using the clustering labels as pseudo-labels for each example. Searching for the best δ, we repeat the process and pick the largest δ so that at least $\alpha = 0.95$ of the balls are pure.

In Fig. 4.1, we plot the percentage of pure balls across different datasets as a function of δ, where the dashed line represents the δ^* chosen by our method.



(a) CIFAR-10 (b) CIFAR-100 (c) Tiny-ImageNet (d) ImageNet

Figure 4.1: Ball purity, as a function of δ, estimated from the unlabeled data (see text). The dashed line marks the highest δ, after which purity drops below $\alpha = 0.95$.

5 Results

In this chapter we present the results of our empirical study, comparing *ProbCover* to other AL strategies in a variety of settings. We focus on the very low budget regime, where the budget size b is of a similar order of magnitude as the number of classes. It is worth noting that in this scenario, the initial labeled set is likely going to be unbalanced across classes, as the data is picked from an unlabeled pool, so in the early stages of training some classes will almost always be missing. Despite this hurdle, *ProbCover* consistently outperforms the competitors and demonstrates its robustness, as seen below.

5.1 Methodology

The following three deep active learning frameworks are evaluated:

(i) *Fully supervised*: A fully supervised DNN with the ResNet-18 architecture is trained using only annotated data.

(ii) *Semi-supervised by transfer learning*: A representation of the data is created by training with a self-supervised task on the unlabeled data, and then a 1-NN classifier is constructed using this representation in a supervised manner. This framework is intended to capture the basic benefits of semi-supervised learning, regardless of the added benefits provided by modern semi-supervised learning methods or the more sophisticated derivation of pseudo-labels.

(iii) *Fully semi-supervised*: A competitive semi-supervised model is trained on both the annotated and unlabeled data. In our experiments we use FlexMatch by Zhang et al. [52].

In frameworks (i) and (ii) we adopt the evaluation kit created by Munjal et al. [33], in which we can compare multiple deep AL strategies in a principled way.

In framework (iii), we adopt the code and hyper-parameters provided by FlexMatch.

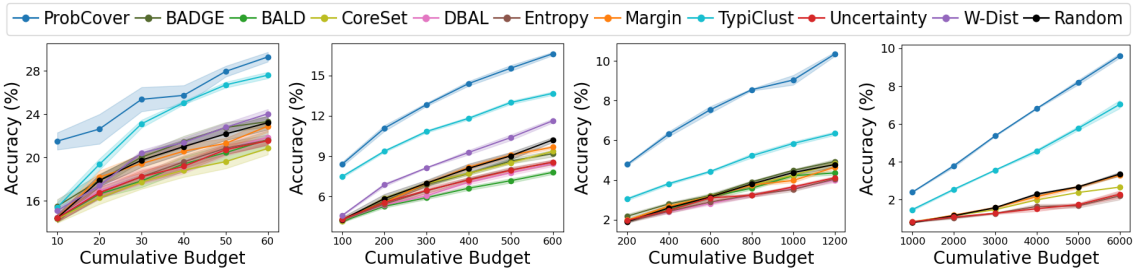
When evaluating frameworks (i) and (ii), we compare *ProbCover* to 9 deep AL strategies as baselines. (1) ***Random*** query uniformly. (2)-(4) query examples with the lowest score, using the following basic scores: (2) ***Uncertainty*** – max softmax output, (3) ***Margin*** – margin between the two highest softmax outputs, (4) ***Entropy*** – inverse entropy of softmax outputs. (5) ***BADGE*** [1]. (6) ***DBAL*** [14]. (7) ***TypiClust*** [18]. (8) ***BALD*** [26]. (9) ***W-Dist*** [30], see also App. B.4. (10) ***Coreset*** [37]. We note that while most baseline methods are suitable for the high budget regime, *TypiClust* and *W-Dist* are also suitable for the low budget regime. Similarly to *ProbCover*, *TypiClust* requires a good embedding space to work properly. When comparing *ProbCover* and *TypiClust*, and in order to avoid possible confounds, we use the same embedding space for both methods.

These AL methods are evaluated on the following classification datasets: CIFAR-10/100 [28], TinyImageNet, [29], ImageNet [12] and its subsets (following Van Gansbeke et al. [44]). When considering CIFAR-10/100 and TinyImageNet, we use as input the embedding of SimCLR [9] across all methods. When considering ImageNet we use as input the embedding of DINO [5] throughout. Results on ImageNet-50/100 are deferred to App. B. Details concerning specific networks and hyper-parameters can be found in App. A, and in the attached code in the supplementary material. When evaluating frameworks (i) and (ii), we perform 5 active learning rounds, querying a fixed budget of b examples in each round. In framework (iii), as FlexMatch is computationally demanding, we only evaluate methods on their initial pool selection capabilities.

5.2 Main Results

(i) Fully supervised framework. We evaluate different AL methods based on the performance of a deep neural network trained directly on the raw queried data. In each round, we query b samples where b is equal to the number of classes in each dataset, and train a ResNet-18 on the accumulated queried set. We repeat this for 5 active learning rounds, and plot the mean accuracy of 5 repetitions (3 for ImageNet) in Fig. 5.1 (see App. B for additional results).

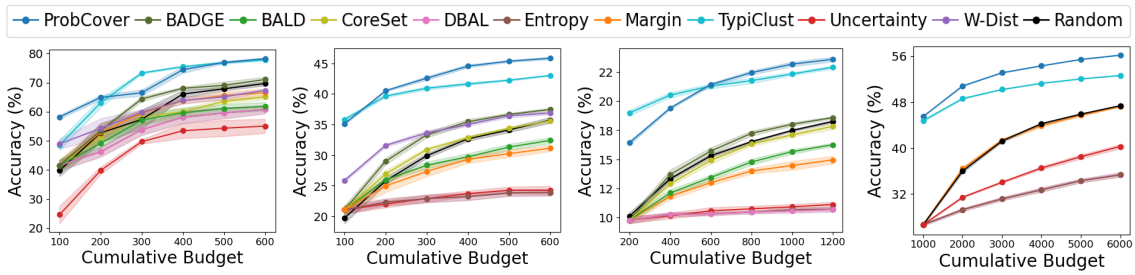
(ii) Semi-supervised by transfer learning. In this framework, we make use of pretrained self-supervised features, and measure classification performance using the 1-NN classifier. Accordingly, each point is classified by the label of its nearest neighbor (within the selected labeled set L) in the self-supervised features



(a) CIFAR-10 (b) CIFAR-100 (c) Tiny-ImageNet (d) ImageNet

Figure 5.1: Framework (i), fully supervised: The performance of *ProbCover* is compared with baseline AL strategies in image classification tasks in the low budget regime. Budget b guarantees on average 1 sample per class, thus the initial sample may be imbalanced. The final average test accuracy in each iteration is reported, using 5 repetitions (3 for ImageNet). The shaded area reflects the standard error across repetitions.

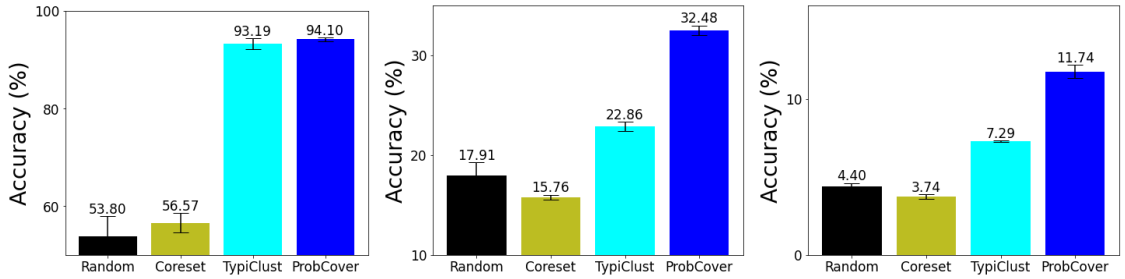
space. In low budgets, this framework outperforms the fully-supervised framework (i), though it is not as effective as the full-blown semi-supervised learning framework (iii). This supports the generality of our findings, not limited to any specific semi-supervised method. Similarly to Fig. 5.1, in Fig. 5.2 we plot the mean accuracy of 5 repetitions for the different tasks.



(a) CIFAR-10 (b) CIFAR-100 (c) Tiny-ImageNet (d) ImageNet

Figure 5.2: Comparative evaluation of framework (ii) - semi-supervised by transfer learning, see caption of Fig. 5.1.

(iii) Semi-supervised framework. We compare the performance of different AL strategies used prior to running FlexMatch, a state-of-the-art semi-supervised method. In Fig. 5.3 we show results with 3 repetitions of FlexMatch, using the labeled sets provided by different AL strategies and budget b equal to the number of classes. We see that *ProbCover* outperforms random sampling and other AL baselines by a large margin. We note that in agreement with previous works [6, 18], AL strategies that are suited for high budgets do not improve the results of random sampling, while AL strategies that are suited for low budgets achieve large improvements.



(a) CIFAR-10 (b) CIFAR-100 (c) Tiny-ImageNet

Figure 5.3: Framework (iii) Semi-supervised: comparison of AL strategies in a semi-supervised task. Each bar shows the mean test accuracy after 3 repetitions of FlexMatch trained using b labeled examples, where b is equal to the number of classes in each task. Error bars denote the standard error.

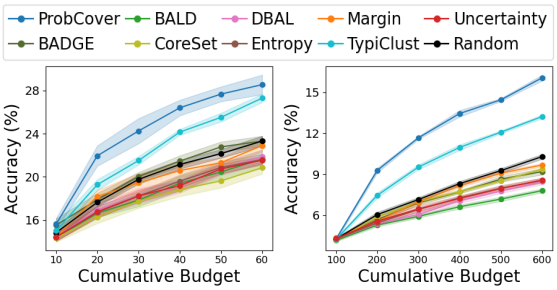
5.3 Ablation Study

We report a set of ablation studies, evaluating the added value of each step of *ProbCover*.

Random initial selection Any method that follows the uncertainty sampling principle requires the model to be minimally trained, which in turn requires a non-empty initial pool of labeled examples. Of all the methods evaluated in the study (see Section 5.1), only two - *ProbCover* and *TypiClust* - are not affected by this limitation. This can be seen in Fig. 5.1, noting that only these two methods do better than random in the initial step. How much of an advantage is conferred by this difference?

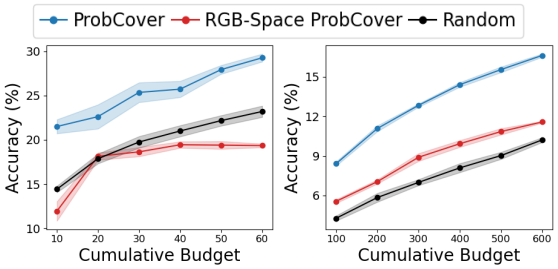
To address this question, we repeat the experiments reported in Fig. 5.1a-5.1b, using an initial random set of annotated examples across the board and by all methods. Results are reported in Fig. 5.4. When comparing Fig. 5.1a-5.1b and Fig. 5.4, we see that the advantage of *ProbCover* and *TypiClust* goes beyond the initial set selection, and remains in effect even if this factor is eliminated.

RGB space distances As discussed in Chapter 4, our approach relies on the existence of a good embedding space, where distance is correlated with semantic similarity. We now verify this claim by repeating the basic fully-supervised experiments (Fig. 5.1) with one difference: *ProbCover* can only use the original RGB space representation to compute distances. Results are shown in Fig. 5.5. When comparing the original *ProbCover* with its variant using RGB space, a significant drop in performance is seen as expected, demonstrating the importance of the semantic embedding space.



(a) CIFAR-10 (b) CIFAR-100

Figure 5.4: Random Initial pool in the supervised framework, an average of 1 sample per class.



(a) CIFAR-10 (b) CIFAR-100

Figure 5.5: Comparison of *ProbCover* when applied to the raw data vs the embedding space.

The interaction between δ and budget size To understand the interaction between the hyper-parameter δ and budget b, we repeat our basic experiments (Fig. 5.1) with different choices of δ and b using CIFAR-10. For each pair (δ, b), we select an initial pool of b examples using *ProbCover* with δ balls, and report the difference in accuracy from the selection of b random points. Average results across 3 repetitions are shown in Fig. 5.6 as a function of b. We see that as the budget b increases, smaller δ's are preferred.

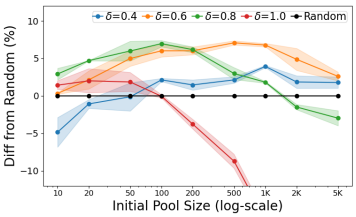
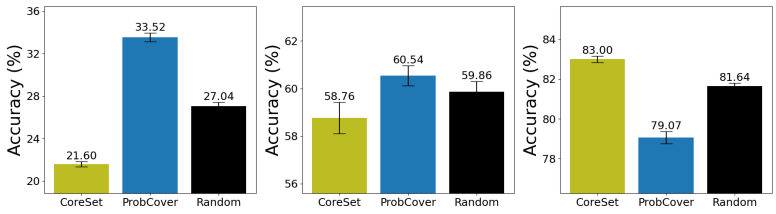


Figure 5.6: The accuracy difference between *ProbCover* when using different δ values, and the outcome of b random samples (average over 3 repetitions).



(a) Low budget (b) Mid budget (c) High Budget

Figure 5.7: Comparing the performance under the supervised framework of *ProbCover* and *Coreset* on different budget regimes. The low budget shows an initial pool selection of 100 samples. Mid/High budget start with 1K/5K samples and query additional 1K/5K samples (see text).

***Coreset* vs. *ProbCover*.** In Section 3.3 we argue that *ProbCover* is suitable for low budgets, while *Coreset* is suitable for high budgets. To verify this claim, we compare their performance under the following 3 setups while using the same embedding space, and report results on CIFAR-10:

- Low budget - Select an initial pool of 100 samples using the SimCLR representation.
- High budget - Train a model on 5K randomly selected examples. Then select

an additional set of 5K examples using the learner's latent representation. This is the setup used by Sener et al. [37].

- Mid budget - Same as high budget, except the initial pool size and added budget are 1K.

Results are reported in Fig. 5.7. In the low budget regime, *ProbCover* outperforms *Coreset* as would be expected. In the mid-budget regime, where the feature space of the learner is informative, only *ProbCover* achieves significant improvement over random selection. In the high budget regime, *Coreset* improves over random selection, while *ProbCover* is least effective.

6 Summary and Discussion

We study the problem of deep active learning in the low-budget regime. We model the problem as *Max Probability Cover*, showing that under certain assumptions on the data distribution, which are likely to hold in self-supervised embedding spaces, it optimizes an upper-bound on the generalization error of a 1-NN classifier. We devise an AL strategy termed *ProbCover* which approximates the optimal solution in a computationally efficient manner.

We empirically evaluate *ProbCover* in supervised and semi-supervised frameworks on the datasets CIFAR-10/100, Tiny-ImageNet and ImageNet. On all datasets, *ProbCover* outperforms the competing strategies, with an improvement of 25-100% over strategies not designed for low budget. This result further underscores the fact that the low-budget regime requires different strategies and approaches.

One drawback of *ProbCover* is the presence of the hyperparameter δ, the radius of the balls. The performance of the method is significantly affected by a good choice of δ, which balances between the notions of *purity* and *coverage*. The choice of δ depends on the label distribution of the data, which is not available in our setting, and is hard to estimate in the low-budget regime. In our experiments, we use a heuristic to sidestep this issue, but there may exist more principled approaches. A better choice of δ will increase the method's performance and robustness on new data and different budget constraints. To that end, we propose several possible avenues of research:

- Inferring a score for δ through the topology of the resulting covering graph. The challenge here is to establish a connection between the graph topology and the label distribution.
- Modifying δ throughout the selection process: currently, *ProbCover* does not fully utilize the "active" aspect of AL. By making use of already known labels, δ can be dynamically adjusted as the querying process progresses.
- Extending the current formulation of *ProbCover*, by making δ sample-dependent rather than uniform. Of course, this turns the problem of settling on one δ

into choosing multiple δs.

- Doing away with δ altogether by considering soft-coverage approaches, where the covering notion is not binary but some continuous measure. The challenges here are choosing an appropriate continuous extension, and addressing the loss of sparsity that binary coverage offers.

Another unrelated line of research is the extension of *ProbCover* to different modalities. Though the research in this thesis was conducted in the context of computer vision, the method we propose requires only a semantic embedding space for the unlabeled samples, so it could be extended to different modalities such as text, where unlabeled data is readily available while labeled data may vary in cost.

Bibliography

[1] Jordan T. Ash et al. "Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[2] Josh Attenberg et al. "Why label when you can search? Alternatives to active learning for applying human resources to build classification models under extreme class imbalance". In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 423–432.

[3] William H Beluch et al. "The power of ensembles for active learning in image classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9368–9377.

[4] Javad Zolfaghari Bengar et al. "Reducing label effort: Self-supervised meets active learning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1631–1639.

[5] Mathilde Caron et al. "Emerging properties in self-supervised vision transformers". In: *arXiv preprint arXiv:2104.14294* (2021).

[6] Yao-Chun Chan et al. "On the Marginal Benefit of Active Learning: Does Self-Supervision Eat Its Cake?" In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3455–3459.

[7] Akshay L Chandra et al. "On initial pools for deep active learning". In: *NeurIPS 2020 Workshop on Pre-registration in Machine Learning*. PMLR, 2021, pp. 14–32.

[8] Liangyu Chen et al. "Making Your First Choice: To Address Cold Start Problem in Vision Active Learning". In: *arXiv preprint arXiv:2210.02442* (2022).

[9] Ting Chen et al. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[10] Ting Chen et al. "Big Self-Supervised Models are Strong Semi-Supervised Learners". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020.

[11] Ekin D Cubuk et al. "Randaugment: Practical automated data augmentation with a reduced search space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 702–703.

[12] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[13] Yarin Gal et al. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[14] Yarin Gal et al. "Deep bayesian active learning with image data". In: *International Conference on Machine Learning*. PMLR, 2017, pp. 1183–1192.

[15] Mingfei Gao et al. "Consistency-based semi-supervised active learning: Towards minimizing labeling cost". In: *European Conference on Computer Vision*. Springer, 2020, pp. 510–526.

[16] Yonatan Geifman et al. "Deep active learning over the long tail". In: *arXiv preprint arXiv:1711.00941* (2017).

[17] Daniel Gissin et al. "Discriminative active learning". In: *arXiv preprint arXiv:1907.06347* (2019).

[18] Guy Hacohen et al. *Active Learning on a Budget: Opposite Strategies Suit High and Low Budgets*. arXiv:2202.02794. type: article. arXiv, Feb. 19, 2022. arXiv: 2202.02794[cs]. URL: http://arxiv.org/abs/2202.02794 (visited on 05/23/2022).

[19] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[20] Tao He et al. "Towards better uncertainty sampling: Active learning with multiple views for deep convolutional neural network". In: *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2019, pp. 1360–1365.

[21] SeulGi Hong et al. "Deep Active Learning with Augmentation-based Consistency Estimation". In: *arXiv preprint arXiv:2011.02666* (2020).

[22] Neil Houlsby et al. "Bayesian active learning for classification and preference learning". In: *arXiv preprint arXiv:1112.5745* (2011).

[23] Neil Houlsby et al. "Cold-start active learning with robust ordinal matrix factorization". In: *International Conference on Machine Learning*. PMLR, 2014, pp. 766–774.

[24] Harry B III Hunt et al. "NC-approximation schemes for NP-and PSPACE-hard problems for geometric graphs". In: *Journal of algorithms* 26.2 (1998). Publisher: Elsevier, pp. 238–274.

[25] Ajay J Joshi et al. "Multi-class active learning for image classification". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 2372–2379.

[26] Andreas Kirsch et al. "Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning". In: *Advances in neural information processing systems* 32 (2019), pp. 7026–7037.

[27] Andreas Krause et al. "Submodular function maximization." In: *Tractability* 3 (2014), pp. 71–104.

[28] Alex Krizhevsky et al. "Learning multiple layers of features from tiny images". In: *Online* (2009). Publisher: Citeseer.

[29] Ya Le et al. "Tiny imagenet visual recognition challenge". In: *CS 231N* 7.7 (2015), p. 3.

[30] Rafid Mahmood et al. "Low Budget Active Learning via Wasserstein Distance: An Integer Programming Approach". In: *arXiv preprint arXiv:2106.02968* (2021).

[31] Dániel Marx. "Efficient approximation schemes for geometric problems?" In: *European Symposium on Algorithms*. Springer, 2005, pp. 448–459.

[32] Sudhanshu Mittal et al. "Parting with illusions about deep active learning". In: *arXiv preprint arXiv:1912.05361* (2019).

[33] Prateek Munjal et al. "Towards Robust and Reproducible Active Learning Using Neural Networks". In: *ArXiv* abs/2002.09564 (2020).

[34] George L Nemhauser et al. "An analysis of approximations for maximizing submodular set functions—I". In: *Mathematical programming* 14.1 (1978). Publisher: Springer, pp. 265–294.

[35] Kossar Pourahmadi et al. "A Simple Baseline for Low-Budget Active Learning". In: *arXiv preprint arXiv:2110.12033* (2021).

[36] Hiranmayi Ranganathan et al. "Deep active learning for image classification". In: *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3934–3938.

[37] Ozan Sener et al. "Active Learning for Convolutional Neural Networks: A Core-Set Approach". In: *International Conference on Learning Representations*. 2018.

[38] Burr Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Cham: Springer International Publishing, 2012. DOI: 10.1007/978-3-031-01560-1. URL: https://link.springer.com/10.1007/978-3-031-01560-1 (visited on 10/20/2022).

[39] Shai Shalev-Shwartz et al. *Understanding Machine Learning: From Theory to Algorithms*. Google-Books-ID: ttJkAwAAQBAJ. Cambridge University Press, May 19, 2014. 415 pp. ISBN: 978-1-107-05713-5.

[40] Changjian Shui et al. "Deep active learning: Unified and principled method for query and training". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 1308–1318.

[41] Oriane Siméoni et al. "Rethinking deep active learning: Using unlabeled data at model training". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 1220–1227.

[42] Samarth Sinha et al. "Variational adversarial active learning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 5972–5981.

[43] Kihyuk Sohn et al. "FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020.

[44] Wouter Van Gansbeke et al. "Scan: Learning to classify images without labels". In: *European Conference on Computer Vision*. Springer, 2020, pp. 268–285.

[45] Zengmao Wang et al. "A batch-mode active learning framework by querying discriminative and representative samples for hyperspectral image classification". In: *Neurocomputing* 179 (2016). Publisher: Elsevier, pp. 88–100.

[46] Daphna Weinshall et al. "Beyond novelty detection: Incongruent events, when general and specific classifiers disagree". In: *Advances in Neural Information Processing Systems* 21 (2008).

[47] Yi Yang et al. "Multi-class active learning by uncertainty sampling with diversity maximization". In: *International Journal of Computer Vision* 113.2 (2015). Publisher: Springer, pp. 113–127.

[48] Changchang Yin et al. "Deep similarity-based batch mode active learning with exploration-exploitation". In: *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017, pp. 575–584.

[49] Donggeun Yoo et al. "Learning loss for active learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 93–102.

[50] Yue Yu et al. "Cold-Start Data Selection for Few-shot Language Model Fine-tuning: A Prompt-Based Uncertainty Propagation Approach". In: *arXiv preprint arXiv:2209.06995* (2022).

[51] Michelle Yuan et al. "Cold-start Active Learning through Self-supervised Language Modeling". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Ed. by Bonnie Webber et al. Association for Computational Linguistics, 2020, pp. 7935–7948.

[52] Bowen Zhang et al. "FlexMatch: Boosting Semi-Supervised Learning with Curriculum Pseudo Labeling". In: *CoRR* abs/2110.08263 (2021). arXiv: 2110.08263.

[53] Chiyuan Zhang et al. "Understanding deep learning (still) requires rethinking generalization". In: *Communications of the ACM* 64.3 (2021). Publisher: ACM New York, NY, USA, pp. 107–115.

[54] Richard Zhang et al. "The unreasonable effectiveness of deep features as a perceptual metric". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

[55] Yu Zhu et al. "Addressing the Item Cold-Start Problem by Attribute-Driven Active Learning". In: *IEEE Trans. Knowl. Data Eng.* 32.4 (2020), pp. 631–644. DOI: 10.1109/TKDE.2019.2891530.

Appendices

A Implementation Details

Source code used in this work is available at the following url:
`https://github.com/avihu111/TypiClust`

A.1 Selection Method

Representation Learning: CIFAR10, CIFAR100, TinyImageNet. To extract semantically meaningful features, we trained SimCLR using the code provided by Van Gansbeke et al. [44] for CIFAR-10, CIFAR-100 and TinyImageNet. Specifically, we used ResNet-18 [19] with an MLP projection layer to a 128-dim vector, trained for 500 epochs. All the training hyper-parameters were identical to those used by SCAN (all details can be found in Van Gansbeke et al. [44]). After training, we used the 512 dimensional penultimate layer as the representation space.

Representation Learning: ImageNet. We extracted features from the official (ViT-S/16) DINO weights pre-trained on ImageNet. We used the L2 normalized penultimate layer for the embedding. All the exact hyper-parameters can be found at Caron et al. [5].

Randomness in *ProbCover* Selections. In order to reduce the correlation between different repetitions using *ProbCover*, we added the following modification to the selection algorithm: instead of taking the node with the highest degree at each iteration, we selected randomly one of the 5 nodes with the highest degree. We verified that both algorithms achieved similar performance, where the deterministic version has slightly better results.

A.2 Fully Supervised Evaluation

We trained a ResNet18 on the labeled set, using the AL comparison framework created by Munjal et al. [33], and following the protocol described in [18] (see details in [18] and the shared code).

A.3 1-NN Classification with Self-Supervised Embeddings

In these experiments, we also used the framework by Munjal et al. [33]. We extracted an embedding similar to § A.1, with which we trained a 1-NN classifier using the default parameters of *scikit-learn*.

A.4 Semi-Supervised Classification

When training FlexMatch [53], we used the AL framework by Zhang et al. [52]. All experiments involved 3 repetitions.

CIFAR-10. We used the standard hyper-parameters used by FlexMatch [53]. Specifically, we trained WideResNet-28 for 400k iterations using the SGD optimizer, with 0.03 learning rate, 64 batch size, 0.9 momentum, 0.0005 weight decay, 2 widen factor, and 0.1 leaky slope. The weak augmentations used are identical to those used in FlexMatch and include random crops and horizontal flips, while the strong augmentations were generated by RandAugment [11].

CIFAR-100. Similar to CIFAR-10, but increasing the widen factor to 8.

TinyImageNet. We trained ResNet-50, for 1.1m iterations. We used an SGD optimizer, with a 0.03 learning rate, 32 batch size, 0.9 momentum, 0.0003 weight decay, and 0.1 leaky slope. The augmentations were similar to those used in FlexMatch.

B Additional Empirical Results

B.1 Improving the greedy approximation

The greedy approximation used in *ProbCover* guarantees $1 - \frac{1}{e}$ approximation to the maximum cover problem. Hunt et al. [24] showed that a polynomial time approximation scheme (PTAS) exists for this problem, suggesting the possibility of better polynomial approximations. However, Marx [31] proved that there is no efficient PTAS to this problem, implying that such polynomial approximations may not be practical. For example, to achieve a $1 - \frac{1}{e}$ approximation using the PTAS suggested in Marx [31] would require $O(n^{100})$ time. Thus, a significantly better approximation than the greedy solution is left for future work. Instead, we improved the greedy algorithm by choosing at each iteration the optimal 2 balls in a greedy way. While this greedy solution achieves a better approximation in theory, in practice we did not see any improvement over the single-ball greedy solution.

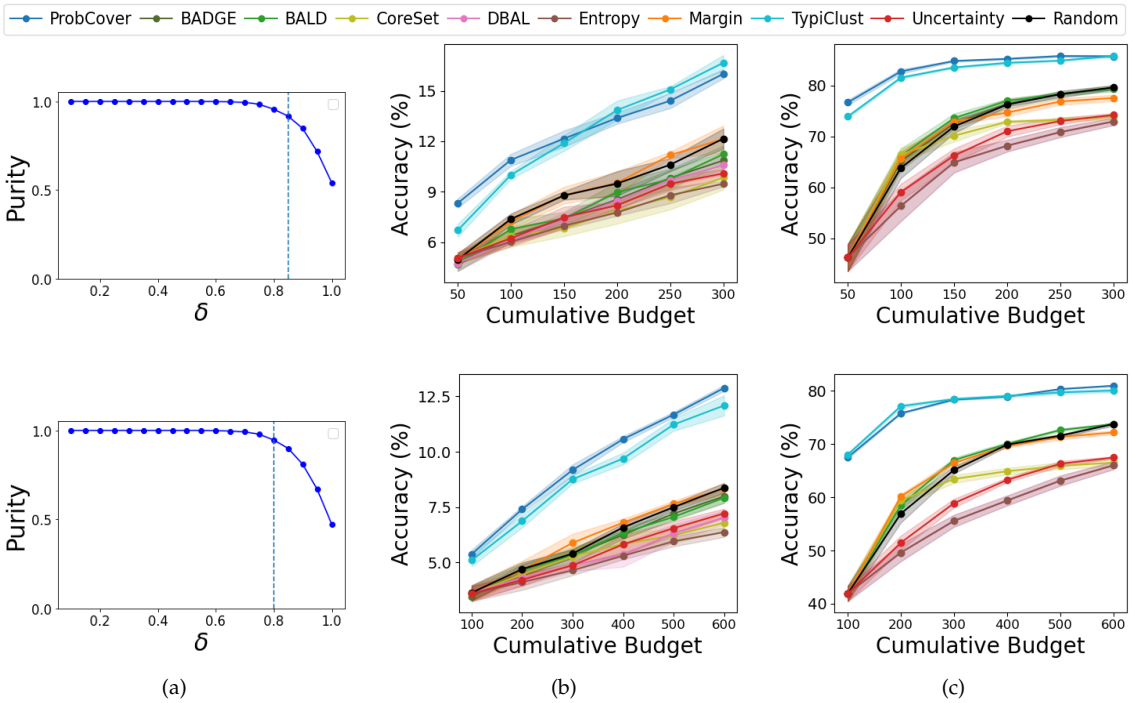
B.2 ImageNet subsets

When evaluating *ProbCover* on ImageNet-50 and ImageNet-100, we report a similar qualitative behavior as seen in other datasets: *ProbCover* performs better than all baselines in the very low-budget regime, using 5 AL rounds with a budget equal to $b = 50$ examples. More concretely, in Fig. 1 we show results corresponding to Figs. 4.1-5.2 when using ImageNet-50.

B.3 TypiClust vs ProbCover on SCAN feature space

Both ProbCover and TypiClust use a unsupervised self-representation embedding as part of an active learning query selection algorithm. In Section 5.2, when comparing *ProbCover* to *TypiClust*, we used the same embedding in both of them, to avoid possible confounds relating to the choice of the specific representation algorithm.

As *TypiClust* reached the best performance using SimCLR representation in most budgets and frameworks on CIFAR-10 and CIFAR-100, we chose that embedding space to compare to *ProbCover*. However, in the fully-supervised framework,



(a) (b) (c)

Figure 1: A Comparative evaluation of *ProbCover* on ImageNet-50 (top row) and ImageNet-100 (bottom row). (a) Similar to Fig 4.1, in which we estimate δ. (b) Similar to Fig. 5.1, trained in the fully supervised framework. (c) Similar to Fig. 5.2, trained in the semi-supervised by transfer learning framework.

with a budget of 10 examples, *TypiClust* yields better results using the embedding space of SCAN.

In Fig. 2, we plot comparison between *ProbCover* and *TypiClust* in this budget, when both are using the embedding space of SCAN. We find a similar trend to the results reported in Section 5.2: *ProbCover* achieves higher accuracy than *TypiClust*, and both surpass random sampling in this budget.

B.4 Comparison with W-Dist

In Chapter 5, we compare *ProbCover* to several other active learning baselines, including W-Dist [30]. As this method is computationally demanding, we are only able to evaluate its performance on the CIFAR-10 and CIFAR-100 datasets, which are the smallest datasets we consider.

We note that the results differ from the results reported in the original paper.

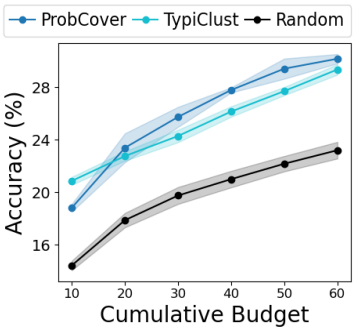


Figure 2: Comparison of *ProbCover* and *TypiClust* using the SCAN feature space. ($\delta = 0.8$)

This stems from several things: firstly, we use 1-NN classification instead of linear classification in the self-supervised scenario. Secondly, the implementation of the Wasserstein method is ours, based on the pseudo-code published in the original work, as no official implementation is available, though we did our best to follow the instructions of the original paper. Thirdly, the method is unique in that it requires a long time to select samples (the original version set a 3 hours timeout for the selection of every 10-20 samples). Instead of the long timeouts suggested in the original work, we used 20 minutes timeouts per round, which reached similar results.