Active Learning Through a Covering Lens

Ofer Yehuda[†], **Avihu Dekel**[†], **Guy Hacohen**[†‡], **Daphna Weinshall**[†]
School of Computer Science & Engineering[†]
Edmond and Lily Safra Center for Brain Sciences[‡]
The Hebrew University of Jerusalem
Jerusalem 91904, Israel
{ofer.yehuda,avihu.dekel,guy.hacohen,daphna}@mail.huji.ac.il

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 nonetheless. Code is available at https://github.com/avihu111/TypiClust.

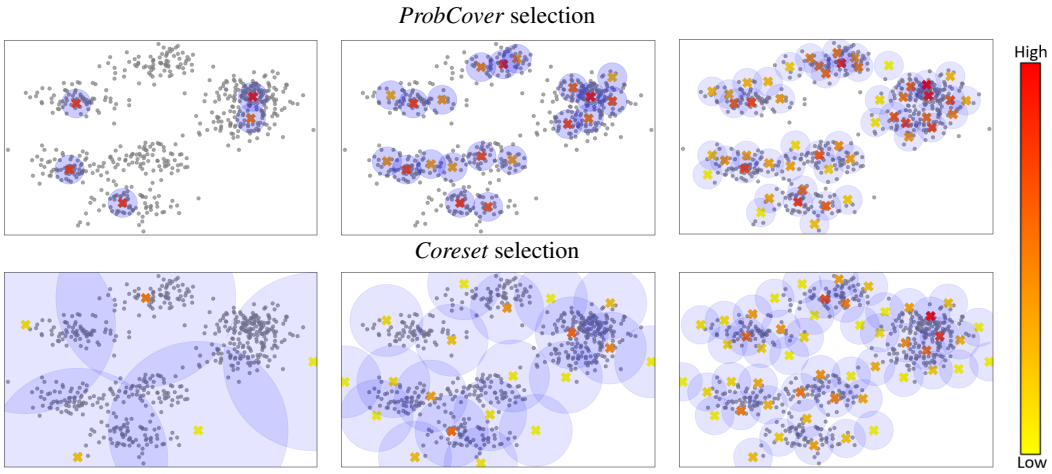
1 Introduction

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 our 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., 25, 47, 3, 13, 14, 35, 24], 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, 37, 17, 39, 36, 16, 46, 41, 45], 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 [34, 38, 7, 31, 53, 20, 2], a phenomenon sometimes termed "cold start" [15, 48, 22]. 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

36th Conference on Neural Information Processing Systems (NeurIPS 2022).



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

Figure 1: *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).

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 anotated data distribution. To face up to this challenge, in Sections 2.1-2.2 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 2.4 we show a connection with existing deep AL methods, like *Coreset* [36], and explain why those methods are more suitable for the high-budget regime than the low-budget regime. This phenomenon is visualized in Fig. 1, 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 [33] (see Section 2.2). To obtain a practical AL strategy, in Section 3 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 Section 4 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.

Relation to prior art Recent work investigated AL methods based on an approximation to a *facility location problem* [44, 29, 52, 36, 42], which is a variant of the covering problem. In the *minimax facility location problem* [12], the entire distribution is covered with a fixed number of balls, which can vary in size, whereas in *ProbCover* the size of the balls is fixed, and we are allowed to cover only part of the total distribution. While this difference may seem minor, in the low-budget regime, when the budget is not large enough to represent the data, examples chosen by the facility location problem are not representative (as illustrated in Fig. 1), which leads to poor performance (as shown in Fig. 10).

Summary of contribution

(i) Develop a theoretical framework to analyze AL strategies in embedding spaces, with "dual" low and high-budget interpretations.

(ii) Introduce *ProbCover*, a low-budget AL strategy motivated by our framework, which significantly outperforms other methods in the low-budget regime.

(iii) Demonstrate the outstanding competence of *ProbCover* in semi-supervised learning with very few labeled examples.

2 Theoretical Analysis

To address the challenge of active learning in low budgets, we adopt a point coverage framework. Computationally, we analyze the generalisation of the Nearest Neighbor (NN) classification model, as this model depends exclusively on distances from a set of training examples and does not involve any additional inductive bias. Thus in Section 2.1 we develop a bound on the generalization error in 1-NN models. It follows from the analysis (see discussion below) that if full coverage is required, which is only practical in the high budget regime, the minimization of this bound translates to the *optimal minimax facility location* problem, which is known to be NP-hard and which the AL *Coreset* algorithm by Sener and Savarese [36] is designed to approximate.

In contrast, in the low budget regime, the aforementioned bound can best be optimized by seeking a labeled set L whose probability of covering the unlabeled set is maximal. In Section 2.2 we show that this problem is also NP-hard. Furthermore, when the data distribution is not known and is being approximated by the empirical distribution, we show that it is equivalent to the classical *Max Coverage* problem. *ProbCover* described in Section 3, is designed to solve this problem. In Section 2.4 we discuss a sense in which the high budget and low budget problems are dual.

2.1 Bounding the Generalization Error

We shall now derive a bound on the generalization error of the 1 Nearest Neighbor (1-NN) classifier. We start with some necessary notations and 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. 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 2.1. We say that a ball $B_\delta(x)$ is *pure* if $\forall x' \in B_\delta(x) : f(x') = f(x)$.

Definition 2.2. We define the *purity* of δ as

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

Notice that $\pi(\delta)$ 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:

$$C_{right} = \{x \in C : \hat{f}(x) = f(x)\}, \qquad C_{wrong} = C \setminus C_{right}.$$

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

Proof. Let $x \in C_{wrong}$. Let $c \in L$ denote the nearest neighbor to x. Then they have the same predicted label, $\hat{f}(x) = \hat{f}(c)$, and $f(c) = \hat{f}(c)$ because c is labeled. Since x is wrongly labeled, $\hat{f}(x) \neq f(x)$, which implies that

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

Finally, since $x \in C_{wrong} \subseteq C$ 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. □

Corollary 1.

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

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

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

Proof.

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

□

Note that (1) gives us a different bound for different δ values, which also depends on the labeled set L. This bound introduces a trade-off: as δ increases, the *coverage* increases, but the *purity* decreases. Ideally, we should seek a pair $\{\delta, L\}$ that achieves the tightest bound.

Discussion We can interpret (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 feasible as we have many points, and the remaining challenge is to reduce $1 - \pi(\delta)$. Accordingly, since $\pi(\delta)$ is monotonically decreasing, we seek to minimize δ subject to the constraint $P(C) = 1$. This is similar to *Coreset* [36]. 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 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*.

2.2 Max Probability Cover

Definition 2.3 (*Max Probability Cover*)**.** Fix $\delta > 0$, and obtain a subset $L \subset X$, $|L| = b$, that maximizes the probability of the covered area $P(C(L, \delta))$

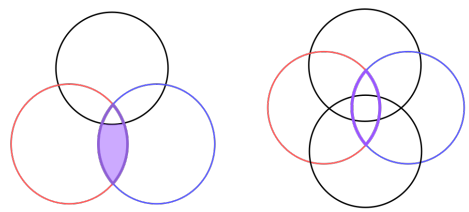
$$\underset{L\subseteq X;|L|=b}{\text{argmax}} \; P(\bigcup_{x\in L} B_\delta(x)) \tag{2}$$

An optimal solution to (2) would minimize the bound in (1), when δ is fixed.

Unfortunately, when moving to practical settings there are two obstacles. The first is complexity:

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

Proof. (Sketch, see full proof in App. B) We construct a reduction from an established NP-hard problem (*Max Coverage*, see Def. A.1) to *Max Probability Cover*. For the collection of subsets $S = \{S_1, \ldots, S_m\}$, we consider the space $\mathbb{R}^m$ and a collection of δ-balls $\{B_\delta(x_i)\}_{i=1}^m$ with the *exhaustive intersection* property. This means that any subset of the balls has at least one point that is contained in all the balls in the subset, but not contained in any other ball (see example in Fig. 2). The existence of such a collection of balls in $\mathbb{R}^m$, $\forall m$, is proved in Lemma 3 (see App. B). We then assign each S_i to $B_\delta(x_i)$, and each element in S_i is mapped to a point in the intersection of all the balls assigned to subsets that contain it. Each such point then defines a Dirac measure, the normalized sum of which determines a probability distribution on $\mathbb{R}^m$. The selection of δ-balls that is the solution to the *Max Probability Cover* can be translated back to a selection of subsets, which is the solution to the original *Max Coverage* problem. □



(a) (b)

Figure 2: Illustration in $\mathbb{R}^2$ of *exhaustive intersection* (see Def. B.2). (a) With 3 balls, every subset of balls has a point that is contained *only* in the specific subset. Thus this set of 3 balls has the *exhaustive intersection* property. (b) With 4 balls, any point in the intersection of two opposite balls is also contained in at least one other ball. Thus this set of 4 balls does **not** have the *exhaustive intersection* property. In the drawing, the region of intersection between the red and blue balls is outlined in purple, while the points within the region that are unique to this pair are marked in light purple. Note that in example (b), this set is empty.

2.3 Using the Empirical Distribution

When employing *Max Probability Cover*, the second practical problem concerns the data distribution, which is hardly ever known apriori. In fact, even when known, the subsequent probabilistic computation is often intractable and hard to approximate. Instead, we may use the *empirical distribution* $\tilde{P}(A) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{x_i \in A}$ as an approximation, which gives us the following useful result:

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

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

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

□

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

The *Coreset* AL method by Sener and Savarese [36] 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 the following loose sense:

1. *Max Probability Cover* minimizes the generalization error bound (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 (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. 1.

3 Method: *ProbCover*

To deliver a practical method, we first note that our approach implicitly relies on the existence of a good embedding space [4, 9, 51], 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., 29, 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 [26], we use the greedy approximate algorithm that achieves $\left(1-\frac{1}{e}\right)$-approximation [26]. A better approximation is impractical, as shown in App. E.1. See App. F for additional time and space complexity analysis.

Below, we describe the greedy algorithm in Section 3.1, and the estimation of ball size δ in Section 3.2.

3.1 Greedy Algorithm

Algorithm 1 *ProbCover*

Input: unlabeled pool U, labeled pool L, budget b, ball-size δ,
Output: a set of points to query
$X \leftarrow$ Embedding of representation learning algorithm on $U \cup L$
$G = (V = X, E = \{(x, x') : x' \in B_\delta(x)\})$
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 $c \in U$ 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*

The algorithm (see Alg. 1 below for pseudo-code) goes as follows: First, construct a directed graph $G = (V, E)$, with $V = X$ the embedding of the data space, and $(x, x') \in E \iff x' \in B_\delta(x) \iff d(x, x') \leq \delta$. 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.

As *ProbCover* uses a sparse representation of the adjacency graph, it is able to scale to large datasets while requiring limited space resources. The complexity analysis of the algorithm, and specifically the complexity of constructing the *adjacency graph* and of the *sample selection*, are discussed in the Appendix F.

3.2 Estimating δ

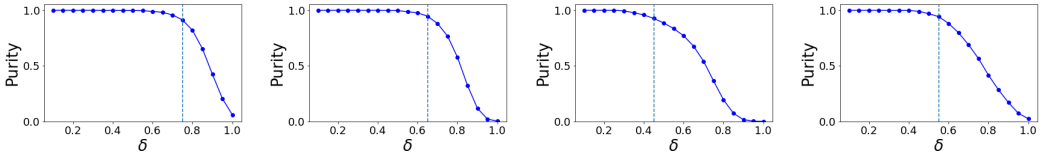
Our algorithm requires the specification of hyper-parameter δ, the ball radius, whose value depends on details of the embedding space (see App. D.1 for embeddings used). In choosing δ, we need to consider the trade-off between large coverage $P(C)$ and high purity $\pi(\delta)$. We resolve this trade-off 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. 3, we plot the percentage of pure balls across different datasets as a function of δ, where the dashed line represents the δ^* chosen by *ProbCover*.



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

Figure 3: 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 $\alpha = 0.95$.

4 Empirical Results

We report a set of empirical results, comparing *ProbCover* to other AL strategies in a variety of settings. We focus on the very low budget regime, with a budget size b of a similar order of magnitude as the number of classes. Note that since the data is picked from an unlabeled pool, chances are that the initial labeled set is not going to be balanced across classes, and in the early stages of training some classes will almost always be missing. *ProbCover*'s excellent performance nevertheless, as seen below, demonstrates its robustness in the presence of this hurdle.

4.1 Methodology

Three deep AL frameworks are evaluated:

(i) *Fully supervised*: train a ResNet-18 only on the annotated data, as a fully supervised task.

(ii) *Semi-supervised by transfer learning*: create a representation of the data by training with a self-supervised task on the unlabeled data, then construct a 1-NN classifier using the ensuing 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 and the more sophisticated derivation of pseudo-labels.

(iii) *Fully semi-supervised*: train a competitive semi-supervised model on both the annotated and unlabeled data. In our experiments we use FlexMatch by Zhang et al. [49].

In frameworks (i) and (ii) we adopt the evaluation kit created by Munjal et al. [32], 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.

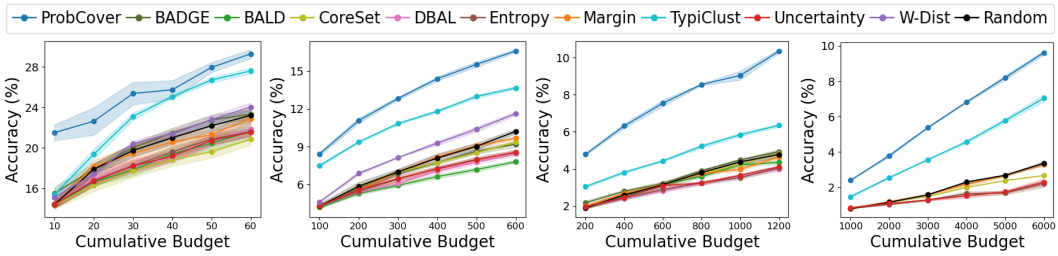
When evaluating frameworks (i) and (ii), we compare *ProbCover* to 9 deep AL strategies as baselines. (1) ***Random*** query uniformly. (2)-(4) query examples with the lowest score, using the following basic scores: (2) ***Uncertainty*** – max softmax output, (3) ***Margin*** – margin between the two highest softmax outputs, (4) ***Entropy*** – inverse entropy of softmax outputs. (5) ***BADGE*** [1]. (6) ***DBAL*** [14]. (7) ***TypiClust*** [18]. (8) ***BALD*** [25]. (9) ***W-Dist*** [29], see also App. E.4. (10) ***Coreset*** [36]. 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 [27], TinyImageNet, [28], ImageNet [11] and its subsets (following Van Gansbeke et al. [40]). When considering CIFAR-10/100 and TinyImageNet, we use as input the embedding of SimCLR [8] 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. E.1. Details concerning specific networks and hyper-parameters can be found in App. D, 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.

4.2 Main Results

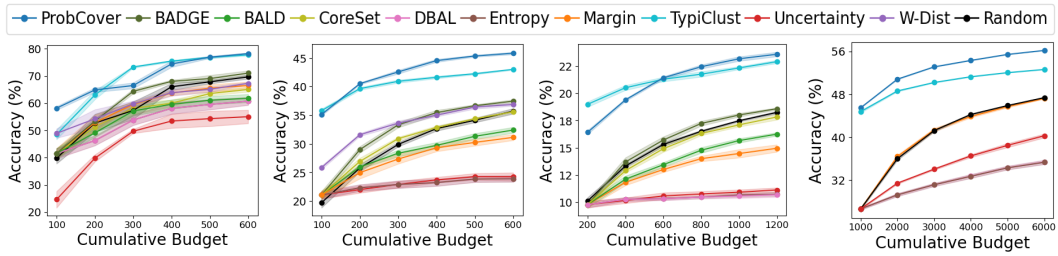
(i) Fully supervised framework. We evaluate different AL methods based on the performance of a deep neural network trained directly on the raw queried data. In each round, we query b samples where b is equal to the number of classes in each dataset, and train a ResNet-18 on the accumulated queried set. We repeat this for 5 active learning rounds, and plot the mean accuracy of 5 repetitions in Fig. 4 (see App. E.1 for additional results).



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

Figure 4: 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.

(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 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. 4, in Fig. 5 we plot the mean accuracy of 5 repetitions for the different tasks.



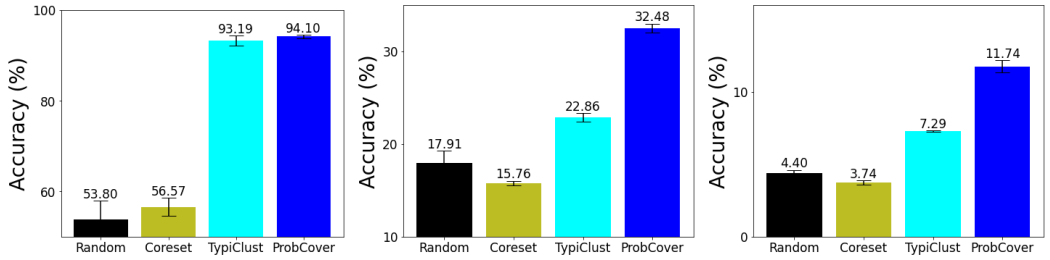
(a) CIFAR-10 (b) CIFAR-100 (c) Tiny-ImageNet (d) ImageNet

Figure 5: Comparative evaluation of framework (ii) - semi-supervised by transfer learning, see caption of Fig. 4.

(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. 6 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.

4.3 Ablation Study

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

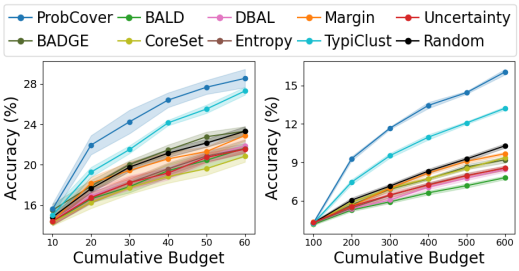


(a) CIFAR-10 (b) CIFAR-100 (c) Tiny-ImageNet

Figure 6: 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.

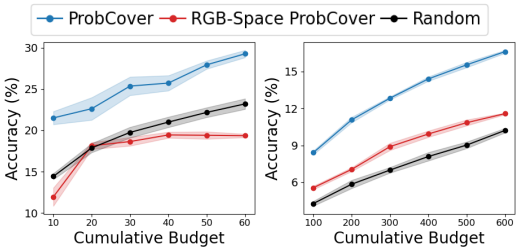
Random initial selection When following the uncertainty sampling principle, as many AL methods do, a trained learner is needed. Any such method requires therefore a non-empty initial pool of labeled examples to train a rudimentary learner, from which uncertainty selection can be bootstrapped. In the set of methods evaluated here (see Section 4.1), only two - *ProbCover* and *TypiClust* - are not affected by this problem. This can be seen in Fig. 4, noting that only these two methods do better than random in the initial step. Is this the only reason they outperform other methods in low budgets?

To address this question, we repeat the experiments reported in Fig. 4a-4b, using an initial random set of annotated examples across the board and by all methods. Results are reported in Fig. 7. When comparing Fig. 4a-4b and Fig. 7, 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.



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

Figure 7: Random Initial pool in the supervised framework, an average of 1 sample per class.



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

Figure 8: Comparison of *ProbCover* when applied to the raw data vs the embedding space.

RGB space distances As discussed in Section 3, 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. 4) with one difference: *ProbCover* can only use the original RGB space representation to compute distances. Results are shown in Fig. 8. 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.

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. 4) 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. 9 as a function of b. We see that as the budget b increases, smaller δ's are preferred.

***Coreset* vs. *ProbCover*.** In Section 2.4 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.

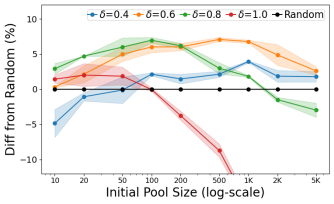
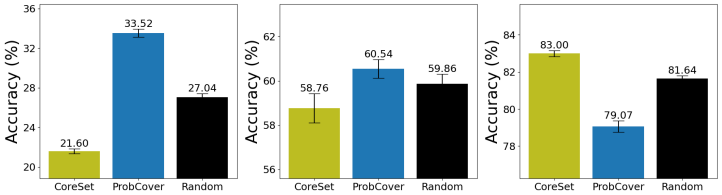


Figure 9: The accuracy difference between *ProbCover* when using different δ values, and the outcome of b random samples (average over 3 repetitions).



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

Figure 10: 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).

- 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 and Savarese [36].
- Mid budget - Same as high budget, except the initial pool size and added budget are 1K.

Results are reported in Fig. 10. 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.

5 Summary and Discussion

We study the problem of AL 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. We empirically evaluate it in supervised and semi-supervised frameworks across different datasets, showing that *ProbCover* significantly outperforms other methods in the low-budget regime.

In future work we intend to investigate: (i) possible avenues for improving the choice of δ by making use of already known labels, or inferring a score for δ through the topology of the resulting covering graph; (ii) extensions of the current formulation of *Max Probability Cover*, by making δ - the radius of the balls - dependent on the samples rather than uniform; (iii) soft-coverage approaches, where the covering notion is not binary but some continuous measure, which may allow us to do away with δ.

Acknowledgments

This work was supported by the Israeli Ministry of Science and Technology, and by the Gatsby Charitable Foundations. We are grateful to our dedicated NeurIPS AC, who acted upon the lengthy discussion between us and the reviewers, as seen on OpenReview.

References

[1] Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 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 and Foster Provost. 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*, pages 423–432, 2010.

[3] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9368–9377, 2018.

[4] Javad Zolfaghari Bengar, Joost van de Weijer, Bartlomiej Twardowski, and Bogdan Raducanu. Reducing label effort: Self-supervised meets active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1631–1639, 2021.

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021.

[6] Yao-Chun Chan, Mingchen Li, and Samet Oymak. 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)*, pages 3455–3459. IEEE, 2021.

[7] Akshay L Chandra, Sai Vikas Desai, Chaitanya Devaguptapu, and Vineeth N Balasubramanian. On initial pools for deep active learning. In *NeurIPS 2020 Workshop on Pre-registration in Machine Learning*, pages 14–32. PMLR, 2021.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[9] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[10] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[12] Reza Zanjirani Farahani and Masoud Hekmatfar. *Facility location: concepts, models, algorithms and case studies*. Springer Science & Business Media, 2009.

[13] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[14] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

[15] Mingfei Gao, Zizhao Zhang, Guo Yu, Sercan Ö Arık, Larry S Davis, and Tomas Pfister. Consistency-based semi-supervised active learning: Towards minimizing labeling cost. In *European Conference on Computer Vision*, pages 510–526. Springer, 2020.

[16] Yonatan Geifman and Ran El-Yaniv. Deep active learning over the long tail. *arXiv preprint arXiv:1711.00941*, 2017.

[17] Daniel Gissin and Shai Shalev-Shwartz. Discriminative active learning. *arXiv preprint arXiv:1907.06347*, 2019.

[18] Guy Hacohen, Avihu Dekel, and Daphna Weinshall. Active learning on a budget: Opposite strategies suit high and low budgets. *arXiv preprint arXiv:2202.02794*, 2022.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Tao He, Xiaoming Jin, Guiguang Ding, Lan Yi, and Chenggang Yan. Towards better uncertainty sampling: Active learning with multiple views for deep convolutional neural network. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1360–1365. IEEE, 2019.

[21] SeulGi Hong, Heonjin Ha, Junmo Kim, and Min-Kook Choi. Deep active learning with augmentation-based consistency estimation. *arXiv preprint arXiv:2011.02666*, 2020.

[22] Neil Houlsby, José Miguel Hernández-Lobato, and Zoubin Ghahramani. Cold-start active learning with robust ordinal matrix factorization. In *International Conference on Machine Learning*, pages 766–774. PMLR, 2014.

[23] Harry B III Hunt, Madhav V Marathe, Venkatesh Radhakrishnan, Shankar S Ravi, Daniel J Rosenkrantz, and Richard E Stearns. Nc-approximation schemes for np-and pspace-hard problems for geometric graphs. *Journal of algorithms*, 26(2):238–274, 1998.

[24] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379. IEEE, 2009.

[25] Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32:7026–7037, 2019.

[26] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability*, 3: 71–104, 2014.

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

[28] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[29] Rafid Mahmood, Sanja Fidler, and Marc T Law. Low budget active learning via wasserstein distance: An integer programming approach. *arXiv preprint arXiv:2106.02968*, 2021.

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

[31] Sudhanshu Mittal, Maxim Tatarchenko, Özgün Çiçek, and Thomas Brox. Parting with illusions about deep active learning. *arXiv preprint arXiv:1912.05361*, 2019.

[32] Prateek Munjal, N. Hayat, Munawar Hayat, J. Sourati, and S. Khan. Towards robust and reproducible active learning using neural networks. *ArXiv*, abs/2002.09564, 2020.

[33] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[34] Kossar Pourahmadi, Parsa Nooralinejad, and Hamed Pirsiavash. A simple baseline for low-budget active learning. *arXiv preprint arXiv:2110.12033*, 2021.

[35] Hiranmayi Ranganathan, Hemanth Venkateswara, Shayok Chakraborty, and Sethuraman Panchanathan. Deep active learning for image classification. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3934–3938. IEEE, 2017.

[36] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

[37] Changjian Shui, Fan Zhou, Christian Gagné, and Boyu Wang. Deep active learning: Unified and principled method for query and training. In *International Conference on Artificial Intelligence and Statistics*, pages 1308–1318. PMLR, 2020.

[38] Oriane Siméoni, Mateusz Budnik, Yannis Avrithis, and Guillaume Gravier. Rethinking deep active learning: Using unlabeled data at model training. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 1220–1227. IEEE, 2021.

[39] Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5972–5981, 2019.

[40] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *European Conference on Computer Vision*, pages 268–285. Springer, 2020.

[41] Zengmao Wang, Bo Du, Lefei Zhang, and Liangpei Zhang. A batch-mode active learning framework by querying discriminative and representative samples for hyperspectral image classification. *Neurocomputing*, 179:88–100, 2016.

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

[43] Daphna Weinshall, Hynek Hermansky, Alon Zweig, Jie Luo, Holly Jimison, Frank Ohl, and Misha Pavel. Beyond novelty detection: Incongruent events, when general and specific classifiers disagree. *Advances in Neural Information Processing Systems*, 21, 2008.

[44] Ziting Wen, Oscar Pizarro, and Stefan Williams. Active self-semi-supervised learning for few labeled samples fast training. *arXiv preprint arXiv:2203.04560*, 2022.

[45] Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G Hauptmann. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113(2):113–127, 2015.

[46] Changchang Yin, Buyue Qian, Shilei Cao, Xiaoyu Li, Jishang Wei, Qinghua Zheng, and Ian Davidson. Deep similarity-based batch mode active learning with exploration-exploitation. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 575–584. IEEE, 2017.

[47] Donggeun Yoo and In So Kweon. Learning loss for active learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 93–102, 2019.

[48] Michelle Yuan, Hsuan-Tien Lin, and Jordan L. Boyd-Graber. Cold-start active learning through self-supervised language modeling. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7935–7948. Association for Computational Linguistics, 2020.

[49] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *CoRR*, abs/2110.08263, 2021.

[50] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3): 107–115, 2021.

[51] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[52] Fedor Zhdanov. Diverse mini-batch active learning. *arXiv preprint arXiv:1901.05954*, 2019.

[53] Yu Zhu, Jinghao Lin, Shibi He, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. Addressing the item cold-start problem by attribute-driven active learning. *IEEE Trans. Knowl. Data Eng.*, 32(4):631–644, 2020. doi: 10.1109/TKDE.2019.2891530.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Throughout the entire paper.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Irrelevant for this work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Section 2.1
   (b) Did you include complete proofs of all theoretical results? [Yes] Section 2, and App. B
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We included the necessary information for reproducibility. Furthermore, the code will be published upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyper-parameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] 4.1
   (b) Did you mention the license of the assets? [Yes] All assets are publicly available.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] The data is publicly available.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] Irrelevant.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Some Definitions

In the proof of the main theorem below we use *Max Coverage*, a known NP-hard problem. We recapitulate its definition as follows:

Definition A.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

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

For convenience, we also repeat the definition of *Max Probability Cover* from Section 2.2:

Definition A.2 (*Max Probability Cover*)**.** Let $\mathbb{X}$ denote the input probability space. Let $X = \{x_i\}_{i=1}^m$, $x_i \in \mathbb{X}$ denote a set of points. Let $b \in \mathbb{N}$ denote an integer, and fix $\delta > 0$. In the problem of *Max Probability Cover* we wish to find a subset $L \subset X$, $|L| = b$, that maximizes the following probability

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

B *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. B.2). We then use this collection in order to construct a reduction from *Max Coverage* to *Max Probability Cover*.

B.1 *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 B.1 (Inversion mapping)**.** We define the *inversion* mapping $\iota : \mathbb{R}^m \setminus \{0\} \to \mathbb{R}^m \setminus \{0\}$ as

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

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

Proof. Let $x \in X_i$. 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. 11a). We use the polarization identity $a \cdot b = \frac{1}{2}(\|a\|^2 + \|b\|^2 - d(a,b)^2)$

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

□

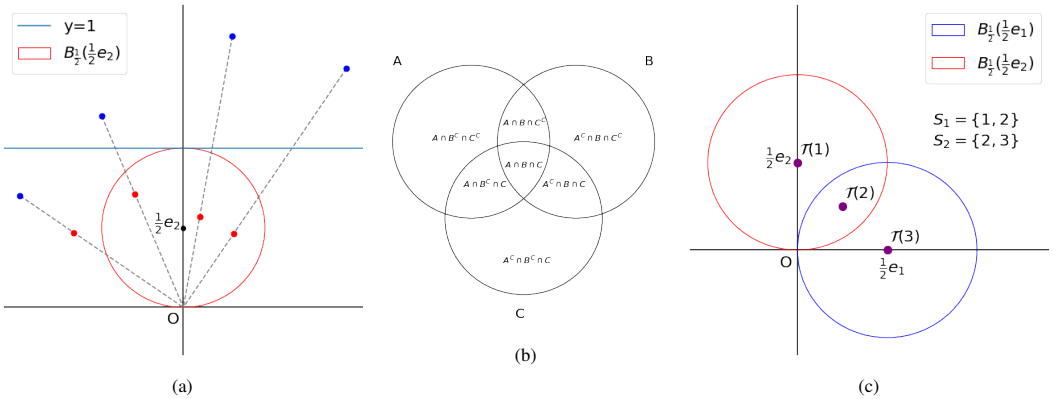


Figure 11: (a) Visualization of Lemma 2. The blue points above the plane $y = 1$ are mapped by $\iota(\cdot)$ to the red points inside $B_{\frac{1}{2}}(\frac{1}{2}e_2)$. Points below $y = 1$ are mapped outside the ball. (b) The exhaustive intersection in $\mathbb{R}^2$. (c) Visualization of the induced 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 $P(B_{\frac{1}{2}}(\frac{1}{2}e_1)) = P(B_{\frac{1}{2}}(\frac{1}{2}e_2)) = \frac{2}{3}$.

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

Definition B.2 (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. 11b).

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. B.2. 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. □

B.2 Reduction from *Max Coverage* to *Max Probability Cover*

To improve readability, we restate in App. A the precise definitions of *Max Probability Cover* (Def. A.2) and *Max Coverage* (Def. A.1). 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. 11c).

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. □

D Implementation Details

Source code used in this work is available at the following url:

https://github.com/avihu111/TypiClust

D.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. [40] 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. [40]). 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.

D.2 Fully Supervised Evaluation

We trained a ResNet18 on the labeled set, using the AL comparison framework created by Munjal et al. [32], and following the protocol described in [18] (see details in [18] and the shared code).

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

In these experiments, we also used the framework by Munjal et al. [32]. We extracted an embedding similar to § D.1, with which we trained a 1-NN classifier using the default parameters of Sklearn.

D.4 Semi-Supervised Classification

When training FlexMatch [50], we used the AL framework by Zhang et al. [49]. All experiments involved 3 repetitions.

CIFAR-10. We used the standard hyper-parameters used by FlexMatch [50]. 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 [10].

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.

E Additional Empirical Results

E.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. [23] showed that a polynomial time approximation scheme (PTAS) exists for this problem, suggesting the possibility of better polynomial approximations. However, Marx [30] 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 [30] 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.

E.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. 12 we show results corresponding to Figs. 3-5 when using ImageNet-50.

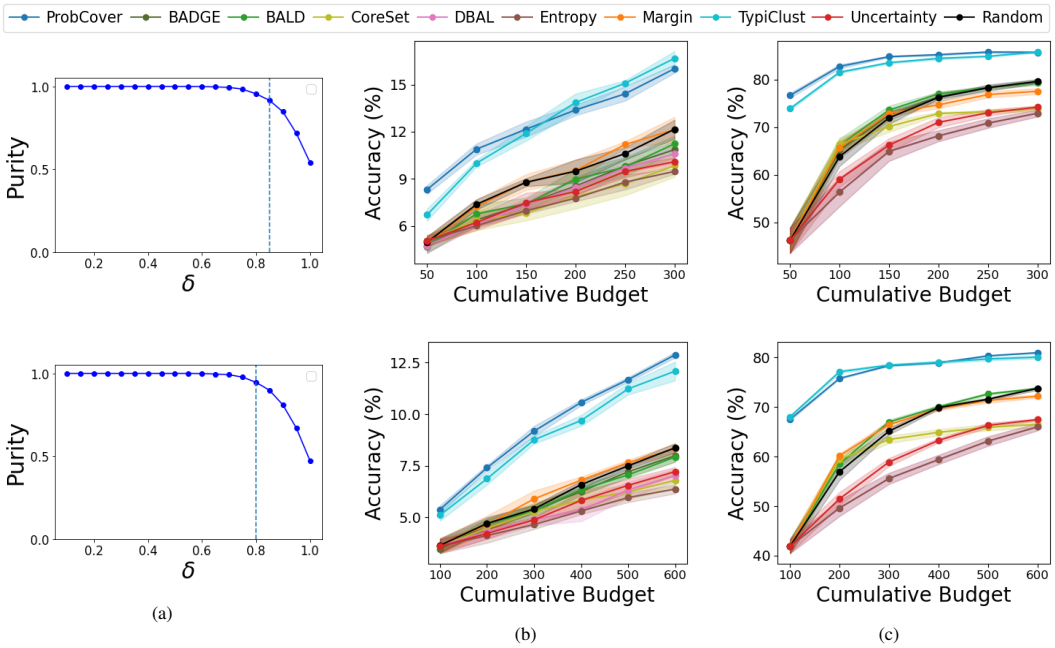


Figure 12: A Comparative evaluation of *ProbCover* on ImageNet-50 (top row) and ImageNet-100 (bottom row). (a) Similar to Fig 3, in which we estimate δ. (b) Similar to Fig. 4, trained in the fully supervised framework. (c) Similar to Fig. 5, trained in the semi-supervised by transfer learning framework.

E.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 4.2, 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, with a budget of 10 examples, *TypiClust* yields better results using the embedding space of SCAN.

In Fig. 13, 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 4.2: *ProbCover* achieves higher accuracy than *TypiClust*, and both surpass random sampling in this budget.

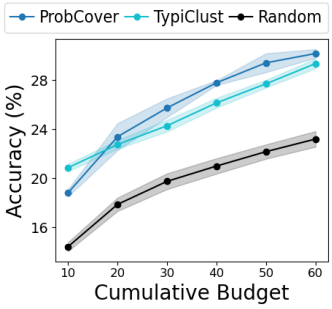


Figure 13: Comparison of *ProbCover* and *TypiClust* using the SCAN feature space. ($\delta = 0.8$)

E.4 Comparison with W-Dist

In Section 4, we compare *ProbCover* to several other active learning baselines, including W-Dist [29]. As this method is computationally demanding, we are only able to evauate 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. 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.

F Time and Space complexity of ProbCover

During the training of a neural network, ProbCover is executed a single time in order to select the best subset to query for human annotation for subsequent network training.

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

F.1 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 samples from ImageNet with $\delta = 0.55$, the average degree was 24 and the algorithm total memory consumption was $12GB$.

F.2 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 $O(n^2 k)$ overall 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 in CIFAR-10/100.