Data Coverage for Detecting Representation Bias in Image Datasets: A Crowdsourcing Approach

Melika Mousavi
mmousa7@uic.edu
University of Illinois Chicago

Nima Shahbazi
nshahb3@uic.edu
University of Illinois Chicago

Abolfazl Asudeh
asudeh@uic.edu
University of Illinois Chicago

ABSTRACT

Existing machine learning models have proven to fail when it comes to their performance for minority groups, mainly due to biases in data. In particular, datasets, especially social data, are often not representative of minorities. In this paper, we consider the problem of representation bias identification on image datasets without explicit attribute values. Using the notion of data coverage for detecting a lack of representation, we develop multiple crowdsourcing approaches. Our main idea, at a high level, is a divide and conquer algorithm that applies a search space pruning strategy to efficiently identify if a dataset misses proper coverage for a given group. We provide a different theoretical analysis of our algorithm, including a tight upper bound on its performance which guarantees its near-optimality. Using this algorithm as the core, we propose multiple heuristics to reduce the coverage detection cost across different cases with multiple intersectional/non-intersectional groups. We demonstrate how the pre-trained predictors are not reliable and hence not sufficient for detecting representation bias in the data. Finally, we adjust our core algorithm to utilize existing models for predicting image group(s) to minimize the coverage identification cost. We conduct extensive experiments, including live experiments on Amazon Mechanical Turk to validate our problem and evaluate our algorithms' performance.

PVLDB Reference Format:
Melika Mousavi, Nima Shahbazi, and Abolfazl Asudeh. Data Coverage for Detecting Representation Bias in Image Datasets: A Crowdsourcing Approach. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://anonymous.4open.science/r/ImageDataCvgCrwd-DE60.

1 INTRODUCTION

Tracing back machine bias to its source, there have been major efforts to identify different types [14, 23, 29] and sources [10, 12, 38] of biases in data. *Representation bias* [33], in particular, happens when a dataset fails to represent some parts of the target population [36]. Lack of representation from certain minority groups in data has caused many instances of machine bias and algorithmic unfairness in data-driven algorithms. For example, on multiple occasions, Google's image search results have been reflecting societal biases, with the most famous example being the "CEO" search query returning only pictures of male CEOs in the top results [20]. Other notable examples include Facebook's ad algorithm excluding women from seeing specific jobs [16], or commercial gender classification systems from Microsoft, IBM, and Face++ that performed up to 35% worse on dark skin women compared to light skin men [8]. Another example is the infamous "Google gorilla" incident [27] where an early image recognition algorithm released by Google had not been trained on enough dark-skinned faces and failed to label black females appropriately. Last but not least, the blink-detection feature of Nikon Cameras misclassified Asian eyes as being closed [31] due to a lack of representation for this group.

Recognizing the potential harms of representation bias, the line of work on data coverage [1–4, 18, 22, 25, 37] has been introduced to ensure proper representation of minority groups in datasets used for decision making and building advanced data science tools. At a high level, a dataset has proper coverage for a given group if it contains at least a certain amount of samples belonging to that group. With many angles to tackle, the problem of identifying and resolving insufficient coverage has been studied for datasets with discrete [3] and continuous [4] attributes populated in single or multiple [22] relations. Additionally, [1, 2] consider ensuring coverage constraints in preprocessing pipelines by rewriting queries into the closest operation so that certain subgroups are sufficiently represented in the downstream tasks.

Despite the extensive work to detect lack of coverage in a given dataset, *existing work is limited to the structured data*, in the form of a table where every row is a tuple of numeric attributes. On the contrary, many of the well-known incidents of representation bias causing machine bias, including all aforementioned examples, are in non-tabular contexts, such as multi-media or textual data.

Admitting the wide range of multimedia databases, with attributes of interest being in different forms and cardinalities, as our first attempt in this project, we consider *image data and a small number of low-cardinality categorical attributes*. Our choices are motivated by image data's popularity in data science tasks and the reported unfairness issues in the image application domains. Our assumption of the attributes of interest follows the fact that sensitive attributes such as `race` and `gender` are low-cardinality and non-ordinal, where each value such as `black` or `female` represents a specific demographic group.

It is common that image datasets usually lack explicit values for attributes of interest (such as `gender` or `race`), crucial for coverage identification. An image dataset is often a collection of images from different domains with little to no information about their domain and which groups they belong to. As a result, even studying

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

coverage over low-cardinality and categorical attributes of interests is challenging in these cases.

There are multiple directions one can seek to overcome such challenges. Considering a small number of categorical attributes of interest (such as `race` and `gender`), one can use off-the-shelf automated techniques, such as classifiers, to first label tuples with their demographic information[1]. Then, relying on the predicted groups, apply the coverage detection techniques to identify the lack of coverage in data. However, as we observe in our experiments, this approach fails, mainly due to the following issues:

(1) *(Machine Bias)*: while the objective of identifying lack of coverage is to minimize machine bias, using (problematic) off-the-shelf models will transfer their biases into the labeled data, causing bias in the evaluation of the dataset. For example, consider a gender-detection classifier. Due to the inherent issues in how the classifier has been trained (and the data it used), it may perform differently across different minority groups. For instance, in our observations (Table 1), we discovered that the precision of a gender classifier from a well-known face recognition framework such as DeepFace [32] for females can get as low as 8% for a given image dataset. These observations along with many real-life examples of classifiers' failure to perform well for minority groups further support the idea that there is no guarantee that relying on the existing models for the purpose of coverage identification leads to precise and robust outcomes.
(2) *(Lack of distribution generalizability)*: Existing tools are trained using data that may come from a different application domain, following a different distribution, and hence may not perform well on the dataset to be evaluated. Let us consider the example of a gender-detection classifier once again. Suppose the classifier has been trained using the standard portray images with a solid background. One cannot expect the classifier to perform well on randomly taken images [21]. Note that applying transfer learning techniques to retrain the model using the dataset to be evaluated is not helpful: we cannot expect a model to identify the lack of coverage issues of the same dataset it is trained on.

Considering the above issues with the existing data-driven tools and techniques, a promising approach to consider is crowdsourcing: to efficiently use human workers to identify a lack of coverage issues. Crowdsourcing is particularly promising for image data, for the tasks such as image labeling, which while being challenging for the machine, are "easy" for human-being to conduct with minimal error. Using crowdsourcing for labeling the images with their attributes of interest can potentially add human bias into the process. Fortunately, accurate and reliable crowdsourcing that minimizes individual errors and biases has been studied well in the literature. Aggregating the responses of multiple crowd workers [5, 11, 17], and profiling the crowd [39–41] are some of the known techniques proposed for this purpose. In addition, we expect human beings to impose minimal bias and error in tasks such as "identifying the gender" of individuals. A baseline solution then can be designed as a two-step process: in the first step, the algorithm asks the crowd to provide the attribute values for all images in the dataset. The second step then applies off-the-shelf coverage identification techniques [3] to detect the uncovered groups.

Cost-effectiveness is a major challenge when using crowdsourcing frameworks such as Amazon Mechanical Turk since there usually is a cost associated with each crowd task. The proposed baseline solution is ineffective in such frameworks because, depending on the size of datasets, it may require a significant number of tasks, meaning a considerable cost to study coverage in a given dataset.

In this paper, therefore, we study the problem of *identifying the lack of coverage in an image dataset with the minimum number of crowd tasks.*

Summary of Contributions. We consider the problem of coverage identification in image datasets using crowdsourcing. In summary, our contributions are the following:

- We propose an efficient divide-and-conquer algorithm to identify the coverage of a demographic group across an image dataset. We provide a tight upper bound on the maximum number of tasks the algorithm generates, and show that it is close to the lower bound on the maximum number of tasks. Using this algorithm as the core, we propose efficient algorithms for coverage identification over different scenarios with multiple non-intersectional and intersectional groups.
- We further introduce practical heuristics for coverage identification by carefully aggregating the minority groups into the so-called "super-groups".
- In presence of pre-trained classification models that predict the group(s) an image belongs to, we adjust our core algorithm to utilize their prediction and minimize the coverage identification cost. In cases where the models accurately predict the group labels, our algorithm only generates a small number of tasks to verify the correctness of the results.
- We evaluate our algorithms using extensive experiments on real and synthetic settings. We run *live experiments on Amazon Mechanical Turk with real workers* to validate our proposal. Besides, our performance evaluation experiments verify our theoretical findings, confirming the effectiveness of our algorithms.

2 PRELIMINARIES

2.1 Data Model

We consider a dataset in form of a collection $\mathcal{D}$ of N objects, each being an image. For example, $\mathcal{D}$ can be a set of $N = 10,000$ human face images. We use the notation t_i to refer to the i-th object in $\mathcal{D}$. We consider the *no explicit attribute-value* model for the data. That is, $\mathcal{D}$ only contains the objects, while the objects are not annotated.

We assume objects are associated with at least one attribute of interest considered for identifying representation bias. Formally, we use $\mathbf{x} = \{x_1, \cdots, x_d\}$ to specify the attributes of interest. Each attribute of interest is a categorical sensitive attribute such as `race`, `gender`, and `age-group`. Each attribute has a cardinality of two or more, specifying different non-overlapping (demographic) groups. For example, a binary attribute `gender` with values `{male, female}` partitions the individuals into two non-overlapping groups.

In particular, we consider three different scenarios with attribute and group models: The most simple scenario is the *single binary*

[1]We would like to underscore that in presence of accurate predictive models, *our algorithms utilize them* to identify coverage with minimum cost to verify the correctness of their results (see § 5).

sensitive attribute case where objects are associated with only one binary sensitive attribute. Many of the problematic representation bias cases that have been reported fall under this category. Examples of this type of attribute include `skin-tone` aggregated into a binary feature of `fair` and `dark` skin-tone [9]. Studies showed that the pulse oximeter devices have a questionable accuracy in measuring arterial oxyhemoglobin saturation in individuals with dark skin-tone [13], which proves it is imperative that skin-tone feature be taken into account in developing this type of device.

The immediate generalization is the *multiple non-intersectional* groups case where each object is associated with one sensitive attribute with cardinality larger than two. A non-binary attribute `race` with values `{White, Black, Hispanic, Asian, Others}`, or a multi-valued attribute `gender` with values `{male, female, non-binary}` are examples of this case.

The next level of generalization is the intersection of multiple attributes where each object is associated with more than one sensitive attribute that can be either binary or non-binary. An example of this case can be the intersection of `race` and `gender`, where each individual can be associated with one value from each of these attributes such as `Asian female` or `Hispanic male`.

2.2 Data Coverage

We use the notion of *data coverage* [3] to identify representation bias in a dataset $\mathcal{D}$. In particular, consider a dataset $\mathcal{D}$ with d attributes of interest $\mathbf{x}$, a count threshold τ (e.g. $\tau = 50$), and a subgroup $\mathbf{g}$ (e.g. `{gender=male, race=white}`) defined over $\mathbf{x}$. The dataset satisfies coverage over $\mathbf{g}$, if there are at least τ objects in $\mathcal{D}$, matching the subgroup $\mathbf{g}$ (e.g. there are more than 50 objects with `gender=male` AND `race=white`).

For datasets with more than one attribute of interest ($d > 1$), *patterns* are used to specify the subgroups. A pattern P is a string of d values, where $P[i]$ is either a value from the domain of x_i, or it is "unspecified", specified with X. For example, consider a dataset with three binary attributes of interest $\mathbf{x} = \{x_1, x_2, x_3\}$. The pattern $P = X01$ specifies all the tuples for which $x_2 = 0$ and $x_3 = 1$ (x_1 can have any value). Consider the universe of all patterns over a set of attributes $\mathbf{x}$. We say a pattern P is a parent of another pattern P' if (a) there exists exactly one attribute x_i on which P and P' are different, while (b) $P[i] = X$ (unspecified). Note that P in this case is a more general subgroup than P', since all the objects matching P' also match P, but the vice-versa is not valid. A pattern P is a *maximal uncovered pattern* (MUP) if there are less than τ objects in $\mathcal{D}$ matching it, while all of its parents are covered. The lack of coverage in a dataset is identified by discovering all of its MUPs.

2.3 Crowdsourcing Model

A major challenge in studying coverage over multimedia data is that the objects are not annotated and we do not know the values on $\mathbf{x}$ beforehand We use crowdsourcing to overcome this challenge. In crowdsourcing platforms such as Amazon Mechanical Turk (AMT) or Crowdflower, a *microtask* or a *Human Intelligence Task (HIT)* is a simple task that usually requires no domain-specific knowledge or expertise, has a clear description and a price which workers can accept and complete and get paid given their result is approved by the requester of the task.

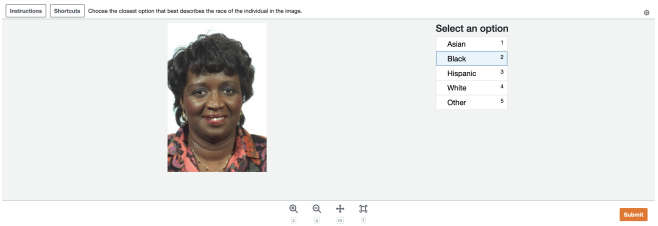


Figure 1: An example of point query to provide a label for `race` attribute

Quality control and aggregation model. Quality of answers is a well-studied crowdsourcing challenge. One popular approach is to employ a redundancy-based strategy in which a single HIT is assigned to multiple workers and the correct answer (the *truth* is inferred by aggregating the multiple answers. There are several studies on truth inference methods. The proposed techniques in this paper are *agnostic* to the choice of the crowdsourcing framework, quality control, and HIT aggregation model. In our experiments, we adopt the popular majority vote strategy [43] to get the truth.

Query/ HIT model. We consider two types of queries:

(1) *Point queries:* A point query is a request to provide a piece of information (attribute value) about a single object. The query itself can be either a yes-no question or providing one or more labels associated with the attributes of interest. In Figure 1 an example of a point query is demonstrated, where the worker is asked to provide the race of an individual.
(2) *Set queries:* While a point query asks the crowd to provide specific attribute values for specific objects, the set queries are for the purpose of *verification*. A set query is a simple yes-no question about a given set of multimedia objects (image, video, etc.). That is to ask if the set contains at least one object belonging to a specific (sub-)group. For example, Figure 2 shows a set query asking the crowd to verify if the set contains any females. In practice, one may need to consider an upper bound on the number of objects in a set query for the query to be reasonable and the answers more accurate.

Pricing model. Pricing and incentive models for crowdsourcing frameworks have been extensively studied. Examples of such methods are *fixed price models*, *bidding models* [34], and *posted price models* [35]. In this paper, we adopt the fixed pricing model, that is all tasks have an equal cost. Therefore, our objective is to minimize the number of tasks required to finish the task of detecting coverage which in turn is in line with minimizing the total cost.

2.4 Problem Definition

Having explained the data model, data coverage, and the crowdsourcing model, we now define our problem as follows:

> **Problem Formulation:** *Given an image dataset $\mathcal{D}$, the attributes of interest $\mathbf{x}$, and the coverage count threshold τ, identify the lack of coverage on $\mathcal{D}$ while minimizing the number of tasks required.*

In particular, we study this problem under three settings: (1) single (minority) group, (2) multiple non-intersectional groups, and (3) intersectional groups, proposing efficient solutions tailored for each setting. At a high level, our main idea is to divide the dataset into subsets of a specified size, ask the crowd a question about

Figure 2: An example of set query about `gender` attribute

the attributes of interest, and based on the crowd's answer to the question, decide to divide that particular subset into two halves or prune it. Modeling each algorithm's flow into a binary tree, we analyze the efficiency of our proposal particularly when the dataset size is very large. We also study the problem in presence of predictive tools for labeling the data, utilizing them to effectively form the tasks, and minimizing the interaction with the crowd.

3 EFFICIENT COVERAGE IDENTIFICATION

We start our technical sections by designing a general algorithm that can be used for detecting coverage over different settings of sensitive attributes. In particular, *given a demographic (sub)group* $\mathbf{g}$*, our goal in this section is to identify if* $\mathbf{g}$ *is uncovered.*

Before proposing the algorithm, however, let us consider the single binary sensitive attribute case to observe a challenge we need to address in our algorithm. In such a setting, the binary attribute partitions the data into two groups: the majority group (to which most of the objects belong) and the minority group. Let $\mathbf{g}_1$ and $\mathbf{g}_2$ be the majority (e.g. `gender=male`) and minority (e.g. `gender=female`) group, respectively.

Challenge. We observe that *verifying that* $\mathbf{g}_1$ *is covered* can be done efficiently. To see why, suppose the coverage threshold is $\tau = 100$, i.e., a group is covered if there are at least 100 instances of it in the dataset. Assume the (majority) group $\mathbf{g}_1$ contains $n_1 \gg 100$ objects in the dataset. In order to verify that $\mathbf{g}_1$ is not uncovered, it is enough for the crowd to discover 100 of those objects, not the entire n_1. Following this, $O(\tau)$ provides a lower bound on the number of crowd tasks required for verifying the coverage for a given group. Still, this lower bound only holds for the groups that are covered, i.e. there at least τ of those in the dataset. Surprisingly, unlike the majority group, verifying that a minority group is indeed uncovered is cumbersome. This is because even though discovering τ objects from a group is enough for verifying that it is covered, one cannot verify a group is uncovered until there is a chance that the dataset might still have enough objects from that group. Thus, assuming a non-zero probability for each unlabeled object to belong to each group, *one might need to ask the crowd to label the entire dataset before one can confirm that a specific group is uncovered.*

3.1 Coverage Identification for a Given Group

Verifying that a minority group is uncovered is challenging. Our idea for addressing this challenge is to design *a divide and conquer algorithm* that, instead of point queries, uses *set queries* to iteratively eliminate subsets of data that *does not include any object from the given group*. At a high level, the algorithm asks a set query from the crowd, inquiring whether the selected set contains at least one object from the given group $\mathbf{g}$ (Figure 2). The user may provide two responses (yes/no). Interestingly, *in either case*, the user response provides useful information that helps in the efficient study of the coverage:

- *The answer is "No":* If the answer to a set query is no, it means the set does not include any object from the given group $\mathbf{g}$. As a result, the algorithm can safely prune the set, asking no further questions about it. In particular, for a group that is not covered, one can expect to see no answers on large set queries helping to quickly prune a large portion of the dataset.
- *The answer is "Yes":* A yes answer to a set query means that the set contains one or more objects from the group $\mathbf{g}$. As a result, the algorithm cannot prune the subset since it can have any number (large than one) of the objects in $\mathbf{g}$. At the first glance, the queries with yes answers do not provide useful information as the algorithm cannot prune the subset (hence it needs to divide it to smaller subsets). However, a key observation is that *the algorithm will only observe a limited number of yes answers* before it stops. The reason is that the number of set queries with yes answers provides a *lower-bound* on the number of objects from $\mathbf{g}$ in the dataset. As a result, as soon as the lower-bound reaches to τ, the algorithm can stop, knowing that $\mathbf{g}$ is covered.

Based on the observations on the answers of the set query tasks, we design our divide and conquer algorithm as follows.

We use a *binary tree* data structure to efficiently implement the algorithm. Each node in the binary tree has the following structure:

```
struct node:
    b_index     // the beginning index of the range
    e_index     // the end index of the range
    parent=null,      // link to the parent node
    left=null,        // link to the left child
    right=null,       // link to the right child
    checked=false, // true if at least one of its
                   // child nodes has returned a yes answer
```

Each node in the tree is associated with a set query containing the objects $\{t_{b\_index}, \cdots, t_{e\_index}\} \in \mathcal{D}$. In addition, every node in the tree has pointers to its parent and children. Finally, every node contains a boolean variable `is_checked` (with the default value `false`) that is used for maintaining a lower-bound on the number of objects discovered from the target group $\mathbf{g}$.

Using the tree data structure, Algorithm 1 shows the pseudo-code of our proposed algorithm for the single binary sensitive attribute case. The algorithm uses the variable `cnt` to maintain the lower-bound number of objects discovered from the target group $\mathbf{g}$. Considering a maximum size of n for the set queries, the algorithm starts by partitioning the data into $\lfloor N/n \rfloor$ subsets, allocating each to a binary tree. Adding the roots of the trees to the queues, the

Algorithm 1 GROUP-COVERAGE

Input: Dataset $\mathcal{D}$, dataset size N, subset size upper bound n, coverage threshold τ, target group $\mathbf{g}$
Output: Coverage of group $\mathbf{g}$

1: $cnt \leftarrow 0$
2: Let Q = an empty queue
3: **for** $i \leftarrow 0$ to N with step size n **do**: `// init roots of subtrees`
4: root $\leftarrow$ node$(i, i+n)$
5: $Q.add(root)$
6: **while** Q is not empty **do**
7: $T \leftarrow Q.del\_top()$
8: $(i, j) \leftarrow (T.b\_index, T.e\_index)$
9: ans $\leftarrow$AskQuestion$(\{t_i, \cdots, t_j\}, \mathbf{g})$ `// set query`
10: **if** T.parent is null **then**
11: **if** ans=yes **then** $cnt \leftarrow cnt + 1$
12: **else continue**
13: **else**
14: **if** ans=no **then**
15: **if** T = T.parent.left **then** $T \leftarrow Q.del(T$.parent.right$)$
16: **else continue**
17: **if** T.parent.checked **then** $cnt \leftarrow cnt + 1$
18: **else** T.parent.checked$\leftarrow$ **true**
19: **if** $cnt = \tau$ **then**
20: **return true** `// coverage threshold satisfied`
21: **if** $j > i$ `/*if setsize>1*/` **then**
22: T.left$\leftarrow$ node$(i, \lfloor \frac{i+j}{2} \rfloor)$; T.right$\leftarrow$ node$(\lfloor \frac{i+j}{2} \rfloor + 1, j)$
23: T.left.prarnt$\leftarrow$ T.right.prarnt$\leftarrow$ T
24: $Q.add(T$.left$)$; $Q.add(T$.right$)$
25: **return false**, cnt `//uncovered`

algorithm then iteratively removes a tree node from the queue until a lower-bound count of τ is achieved for the minority group $\mathbf{g}$ or it verifies that $\mathbf{g}$ is uncovered.

For every node T removed from the queue, the algorithm asks the crowd to verify if its corresponding set contains at least one instance belonging to $\mathbf{g}$. Depending on the answer from the crowd, multiple situations can happen. If the node stands for the root of a binary tree and the response is yes, the algorithm has found at least one more object from $\mathbf{g}$; but if the response is no, it can safely prune the entire set from the search space and continue with other sets. On the other hand, if T is not a root node, if the answer is no, it means the answer for the other child of its parents should be yes. That is because the parent node contained at least one object from $\mathbf{g}$. Therefore, since T does not contain any such object, the other child of its parent should contain at least one. As a result, in Line 15, the algorithm replaces T with the other child of its parents, safely knowing the answer to a query to the new T is yes.

When the answer to a non-root node query is yes, the algorithm may or may not be able to increase the lower-bound cnt. Note that the algorithm has already associated at least one object from $\mathbf{g}$ to the parent of each non-root node (the nodes with no answers have been pruned). As a result, the lower bound on $\mathbf{g}$ gets increased if the answer to both children of a parent node is yes. We use the variable checked for this purpose. checked is true, if the answer to one of the children of a node is yes. Using this, when receiving a yes response, the algorithm increases the lower bound (Line 17) only if the checked variable of the parent of T is true. At any moment that the lower-bound reached the threshold τ, the algorithm stops marking $\mathbf{g}$ as covered. Finally, the algorithm breaks the yes nodes with set sizes larger than one in two halves, adding them to the queue. If after checking all nodes in the queue the threshold τ is not reached, $\mathbf{g}$ is uncovered.

3.2 Algorithm Analysis

LEMMA 1. *(Correctness) The* GROUP-COVERAGE *algorithm successfully identifies if a group* $\mathbf{g}$ *is covered or not, i.e., if there are at least* τ *instances of* $\mathbf{g}$ *in the dataset* $\mathcal{D}$.[2]

Having shown the correctness of the GROUP-COVERAGE algorithm in Lemma 1, we next study the number of tasks generated by it. Before studying the performance in general cases, let us consider two extreme cases, while assuming $N = n$, where (Case I) the answer to all questions is yes, and (Case II) there exists only one object in $\mathbf{g}$.

Case I: Consider the cases where the answer to all set questions is yes, meaning that all set queries contain at least one object belonging to $\mathbf{g}$. In this case, the set queries do not help prune the search space. As a result, the execution tree is a complete binary tree shown in Figure 3b. Note that the number of leaf nodes in this tree shows the value of cnt, knowing that the answers to those set queries are yes. On the other hand, the GROUP-COVERAGE algorithm stops when $cnt = \tau$. Therefore, the number of leaf nodes in the tree is at most τ. In a complete binary tree with τ leaves, there are $\tau - 1$ non-leaf nodes. As a result, the total number of tasks generated by the GROUP-COVERAGE algorithm, in this case, is $2\tau - 1 = O(\tau)$.

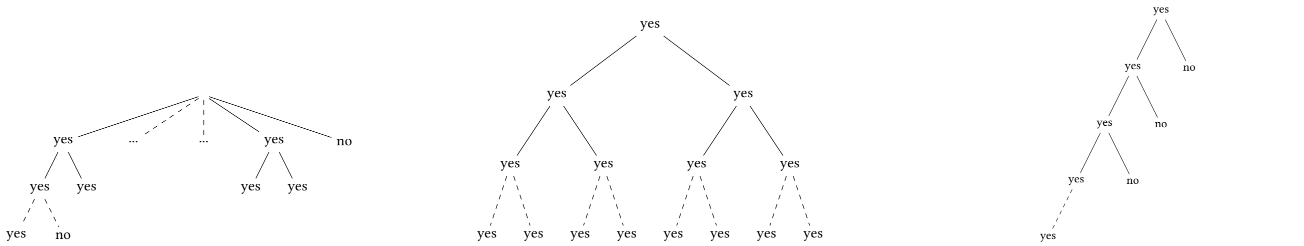
Case II: Suppose there exists only one object from $\mathbf{g}$ in $\mathcal{D}$. As shown in Figure 3c, in this case, every level of the execution tree contains exactly one node while one of its children is no, while the other one is yes. Given that the root of the tree corresponds to a set of n objects, the leaf yes node (containing one object) is at depth $\log n$ in the tree. Therefore, since every intermediate node has exactly two children, the number of tasks generated in this case is $O(\log n)$.

THEOREM 2. *Assuming the dataset size is* $N = n$*, or if there is no upper-bound on the size of set queries:*

- *the maximum number of tasks generated by the* GROUP-COVERAGE *algorithm is* $O(\tau \log n)$.
- *the upper bound is tight.*

PROOF. We first prove the upper bound. The algorithm prunes the sets associated with no answers. Let us assign each no leaf node to its immediate non-leaf parent in the execution tree. As a result, the response to the root node itself is no, the number of no leaf nodes is bounded by the number of non-leaf nodes in the tree. On the other hand, given that the tree does not extend beneath the no nodes, each non-leaf node is the path from a yes leaf node to the root. Hence, the set of non-leaf nodes can be computed as the union of the paths from all yes leaf nodes to the root. The number of yes leaf-node is bounded by τ. Besides, as observed in Case II (Figure 3c) a path from the root of each leaf node is of length at most $\log n$. Therefore, using the *union bound* [26]-Chapter 3.1, the total number of nodes in the execution tree is at most $O(\tau \log n)$.

[2]Due to the space limitations, the proof has been provided in the technical report [6].



(a) The binary tree representation of the algorithms' flow

(b) At least one object belonging to g in all set queries

(c) Only one object belonging to g in $\mathcal{D}$

Figure 3: Various outcomes of the binary tree of the generated tasks

We use an adversarial example to show that the bound is tight. Consider a case where $\mathbf{g}$ is uncovered, there are exactly $\tau - 1$ objects belonging to $\mathbf{g}$, and those are uniformly distributed across the dataset such that the answer to the first $2\tau - 3$ set queries is yes. This is similar to Figure 3b except that there are $\tau - 1$ nodes in level $\log \tau$. At this point, since corresponding sets for each of the $\tau - 1$ node at level $\log \tau$ contain exactly one object belonging to τ, the subtree beneath each node grows similar to Figure 3c until sets of the yes nodes are of size one. The depth of the subtrees beneath each of the $\tau - 1$ nodes in level $\log \tau$ is $\log n - log\tau = \log \frac{n}{\tau}$, each containing $O(\log \frac{n}{\tau})$ nodes. As a result, the total number of nodes in the execution tree is $O(\tau \log \frac{n}{\tau})$ showing that the $O(\tau \log n)$ upper bound is tight. □

Lemma 3. *(Cost Analysis) The maximum number of tasks generated by the* Group-Coverage *algorithm is* $O(\frac{N}{n} + \tau \log n)$.

Proof. Given the upper-bound size $n < N$, the algorithm needs to partition $\mathcal{D}$ into multiple sets queries, each of size n, as in the first level of Figure 3a. The algorithm stops as soon as it finds τ queries with yes answers. In the worst case, however, it may require asking all levels of the $\frac{N}{n}$ set queries at level 1. Suppose the number of level-1 set queries with yes answer is less than τ. In that case, let $r_1, \cdots, r_t$, where $t < \tau$, be the level-1 yes queries – each being the root of the sub-tree beneath them. As argued in the proof of Theorem 2, since (a) there always are at most τ leaf nodes with yes answers, (b) the number of no leaf nodes is bounded by the number of intermediate nodes, (c) the set of all intermediate node in all sub-trees can be represented as the union of the paths from each yes leaf to its corresponding root, and (d) the length of each path is at most $\log n$, the maximum number of nodes in all sub-trees is $O(\tau \log n)$. Therefore, the maximum size of the execution tree is $O(\frac{N}{n} + \tau \log n)$. □

Before concluding this section we would like to note that lower-bound on the maximum number of set queries an algorithm need to issue for deciding if $\mathbf{g}$ is covered is $\frac{N}{n}$. That simply is because $\frac{N}{n}$ queries are needed to include each object in $\mathcal{D}$ in at least one set query. In cases where $\mathbf{g}$ is uncovered, all objects should be queried to verify $\mathbf{g}$ is indeed uncovered. Comparing this lower-bound with the maximum number of tasks generated by the Group-Coverage algorithm, one can verify that Group-Coverage has only a small *additive* overhead of $O(\tau \log n)$ from the lower-bound.

3.3 Coverage Identification using Group-Coverage

So far in this section, we designed an algorithm that, given a group $\mathbf{g}$, efficiently interacts with the crowd to identify if $\mathbf{g}$ is covered. This algorithm can be applied for coverage identification for different scenarios of sensitive attributes, as explained in the following.

3.3.1 Multiple Non-intersectional Groups. In a case where there is one attribute-of-interest x, let c be the cardinality of x. In this case, each value of x specifies a group $\mathbf{g}_i$. To study coverage in this case, one needs to identify if each of the groups $\mathbf{g}_i$ is covered in $\mathcal{D}$. This can simply be done by running the Group-Coverage algorithm c times, where the i-th run checks if the group $\mathbf{g}_i$ is covered. As a result, following Lemma 3, the maximum number of tasks generated for this case is then $O\big(c(\frac{N}{n} + \tau \log n)\big)$.

3.3.2 Intersectional Groups. For cases with intersectional groups, the intersections of attribute values on $\mathbf{x} = \{x_1, \cdots, x_d\}$ specify different subgroups, while the objective is to identify the uncovered region in form of *maximal uncovered patterns (MUPs)* [3]. Consider the graph representation of patterns associated with the attributes $\mathbf{x}$. A real-life example of this case is represented with `gender` and `race` attributes in Figure 4. The first level of the graph contains patterns, such as `female-X` or `X-black` with one attribute-value specified for them. A child of a node at level ℓ is a node at level $\ell + 1$ with one more attribute-value specified than its parent(s). For example, the level-2 pattern `female-black` (representing the subgroup of `female black` individuals) is a child of patterns `female-X` and `X-black`. An MUP is a pattern that is uncovered itself but all of its parents are covered. For example, `female-black` is an MUP if it is uncovered but both its parents `female-X` and `X-black` are covered.

In order to specify MUPs using the Group-Coverage algorithm, we consider the fully-specified subgroups (at maximum level), each being the intersection of d attribute values. The Cartesian product of the values of x_1 to x_d determine these subgroups. For example, in Figure 4 the nodes at level 2 of the graph (e.g. `female-asian`) show the fully-specified subgroups. Using $c_1, \cdots, c_d$ as the cardinality of the attributes $x_1, \cdots, x_d$, the number of fully-specified subgroups is $m = c_1 \times c_2 \times \cdots \times c_d$. Let $\mathbf{g}_1, \cdots, \mathbf{g}_m$ be the set of these subgroups. Finding the MUPs is then possible by combing the Pattern-combiner algorithm [3] with the algorithm Group-Coverage. At a high level, the pattern-combiner starts from the bottom of the pattern graph (fully-specified subgroups), and counts the coverage for each of those, eliminating the covered nodes (with all of their parents). Then moving up in the graph, for the nodes that have not been pruned, it uses the counts of their children to check if those are covered, pruning the covered nodes and their parents while identifying the MUPs (uncovered nodes that all of their parents are pruned).

Algorithm 1 enables running pattern-combiner noting that Group-Coverage finds the *exact count* for an uncovered group. In such cases the lower-bound variable (*cnt* in Algorithm 1) contains the

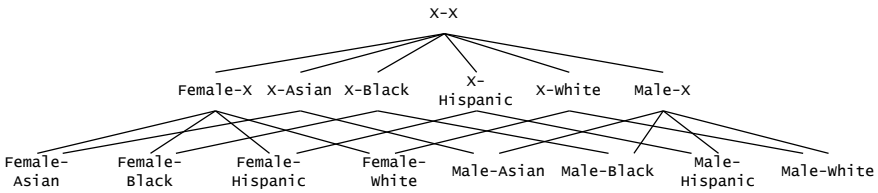


Figure 4: The pattern graph for $\mathbf{x}_1$=gender and $\mathbf{x}_2$=race

number of items belonging to the uncovered group. Additionally, pattern-combiner only needs the counts for the fully-specified subgroups that are uncovered as it already prunes the covered nodes.

4 PRACTICAL OPTIMIZATIONS BASED ON GROUP AGGREGATION

So far in previous sections, we designed an efficient algorithm to detect if a certain group is covered in $\mathcal{D}$. We further explained how this algorithm enables coverage identification for different cases of attributes of interest. In this section, we propose practical heuristics for cases with multiple non-intersectional and intersectional groups. In particular, we note that independently running the Group-Coverage algorithm for different groups specified by $\mathbf{x}$ to identify coverage, misses the opportunity to reuse the information collected during each run.

First, to avoid labeling objects multiple times we move the labeled objects from the unlabeled set $\mathcal{D}$ to the labeled set $\mathcal{L}$. Our main idea, however, is to form set queries that combine multiple demographic groups in one task – instead of one group. Specifically, consider a case where two or more demographic groups in the dataset are uncovered and there are very few items corresponding to them such that after combining them to one "*super-group*", the result is still uncovered. For example, suppose the races `Native American`, `Asian`, and `Middle Eastern` are on the absolute minority that the summation of counts for all these three groups is less than the coverage threshold. In this case, instead of running the Group-Coverage algorithm once for each of them, we can run it for the super-group once. To do so, we change the set query to combine the individual groups with *OR* predicate. The question then, for example, will be "`Is there an individual in this set of images who is either Native American, Asian, or Middle Eastern?`".

Nevertheless, the challenge is that we do not have prior knowledge about the dataset $\mathcal{D}$ to form the super-groups. In order to obtain such information, we consider *estimating the counts using sampling*. To do so, we add a sampling phase at the beginning of our method, in which a small random subset of the dataset is presented to the crowd as *point queries*, with their task being to label the items. The results from this step can give us an estimation of how various demographic groups are represented in the dataset. Based on this estimation, the algorithm will decide which groups to aggregate as super-groups. The next question is how big the sample set should be. How big of a sample do we need? The intuition we use is that point queries are efficient for detecting a majority group as we are expected to detect that the group is covered after $\Theta(\tau)$ point queries. As a result, considering the point queries for detecting that the majority groups, we can *piggyback* on the point query results to collect information about minorities and form the super-groups.

Following this idea, we consider labeling a random subset of size $c\tau$ of $\mathcal{D}$ at the beginning of the algorithm, where c is a small constant (we found $c = 2$ as a good choice in our experiments). Note that in cases where initial point queries do not find at least τ objects from the majority group(s), the algorithm effectively identifies a subset of these instances and needs fewer queries to get to the coverage threshold. As a result, Group-Coverage rapidly stops detecting them as covered.

Algorithm 2 Multiple-Coverage

Input: Dataset $\mathcal{D}$, dataset size N, subset size upper bound n, coverage threshold τ, target groups $\mathbb{G}$
Output: Coverage of all groups in $\mathbb{G}$

$\mathcal{D}, \mathcal{L} \leftarrow$ LabelSample $(\mathcal{D}, \tau)$
$\mathbb{G}_{agg} \leftarrow$Aggregate$(\mathcal{L}, \tau, \mathbb{G})$
for $\mathcal{G} \in \mathbb{G}_{agg}$ **do**:
  $\tau' \leftarrow \tau - \sum_{\mathbf{g}\in\mathcal{G}} \mathcal{L}$.count(g)
  $cnt, cvg \leftarrow$Group-Coverage $(\mathcal{D}, N, n, \tau', \mathcal{G})$
  if $|\mathcal{G}| = 1$ **then** cov.add $(\langle \mathcal{G}, cvg\rangle)$; **continue**
  if cvg = **true then** `// if` $\mathcal{G}$ `is a super-group and covered`
    for $\mathbf{g} \in \mathcal{G}$ **do**:
      $\tau' \leftarrow \tau - \mathcal{L}$.count(g)
      $cnt, cvg \leftarrow$ Group-Coverage $(\mathcal{D}, N, n, \tau', \mathbf{g})$
      cov.add $(\langle \mathbf{g}, cvg, cnt\rangle)$
  else for $\mathbf{g} \in \mathcal{G}$ **do:** cov.add $(\langle \mathbf{g}, \textbf{false}, cnt\rangle)$
return cov

One drawback of forming the super-groups is when the result for a set of super-group is *covered*. In this case, we could not know whether one, two, or all groups are covered, and thus, we need to examine each separately. In other words, the aggregation strategy will incur a penalty cost when the super-groups are covered.

Let $\mathbb{G}$ be the list of groups in one attribute or the set of fully-specified subgroups in the intersection of multiple attributes. We propose Algorithm 6 to form the super-groups.[3]

Aggregation. The count estimations based on the samples collected in the labeled set $\mathcal{L}$ are utilized to set up the super-groups. We calculate the expected number of instances corresponding to each group in the dataset based on their occurrence in the sample. Let $\mathcal{L}$.count(g) return the number of objects in group g that belong to $\mathcal{L}$. Since $\mathcal{L}$ is a random sample of $\mathcal{D}$, the expected number of instances of g in $\mathcal{D}$ is $\mathbb{E}(\mathbf{g}) = N(\mathcal{L}.\text{count}(\mathbf{g}))/|\mathcal{L}|$.

If the expected number is less than the coverage threshold, it is likely that this particular group is uncovered in the dataset and vice versa. Similarly, if the summation of the expected numbers for a set of groups is still less than τ, the super-group formed by merging them is expected to be uncovered. To use this idea for forming the super-groups, the aggregate function in Algorithm 6 first sorts the groups based on their count values in $\mathcal{L}$ ascending. This helps to

[3]Due to the space limitations, the algorithm has been provided in the technical report [6].

put the minority groups nearby and merge them as super-groups. Then it makes a pass over the sorted groups while maintaining the sum over their expected coverage. So far as the expected sums are less than τ, the algorithm keeps merging the groups into a super-group, and then it moves to the next super-group. It finally returns the list $\mathbb{G}_{agg}$ of the super-groups.

Multiple and Intersectional Groups Coverage. Using the idea of merging minority groups into super-groups, Algorithm 2 specifies the uncovered groups for the cases where there exist multiple, non-intersectional groups in a single attribute. In particular, for every group $\mathcal{G}$ in the set of aggregated groups $\mathbb{G}_{agg}$, the algorithm first specifies the number τ' of instances it needs to observe before it can conclude $\mathcal{G}$ is covered. Next it runs the Group-Coverage algorithm for identifying the coverage of $\mathcal{G}$ in $\mathcal{D}$. If $\mathcal{G}$ is not a super-group, the algorithm directly adds the coverage result of $\mathcal{G}$ to the output. For cases where $\mathcal{G}$ is a super-group, if $\mathcal{G}$ is covered, the algorithm fails to conclude if groups $\mathbf{g} \in \mathcal{G}$ are covered or not. Therefore, it reruns the Group-Coverage algorithm for all of such individual groups $\mathbf{g}$. On the other hand, for cases where the super-group $\mathcal{G}$ is uncovered, the algorithm concludes that all groups in $\mathcal{G}$ are uncovered.

We can take advantage of the above technique for multiple attributes, where we are interested in identifying the coverage of each of the individual and the intersectional groups. Figure 4 demonstrates the pattern graph for two attributes of `gender` and `race`. To solve the problem for this case, we take on a similar idea to the Pattern-Combiner algorithm[3]. The objective of the Pattern-Combiner algorithm is to find MUPs (maximal uncovered patterns) in a dataset. As mentioned before, an MUP is a pattern that is uncovered but all of its parents are covered. Consequently, all of the children of MUP are uncovered as well. For example, in the `race` and `gender` attributes case with $\tau = 50$, assuming that we find 15 instances of `Asian-Female` and 20 instances of `Asian-Male`, we can conclude that `Asian` group with total instances of 35 is uncovered as well. On the other hand, if we identify a collective sum of instances for a specific group with its respective subgroups that is greater than or equal to the coverage threshold, we can detect its coverage without the need to apply the Group-Coverage algorithm or additional crowdsourced tasks.

We use this idea to reduce the problem of identifying the coverage of multiple attributes to identifying the coverage of the fully-specified subgroups at the maximum level. We can see that this problem can be easily transformed into solving it for multiple, non-intersectional groups. It is noteworthy that the aggregation process for this special case requires that only the nodes with the same parent be aggregated with each other. To this end, we used a flag (*multi*) in our aggregation algorithm to distinguish between the two cases. Having identified the coverage of the subgroups using the Multiple-Coverage algorithm, we then proceed to identify the coverage of all other patterns in the upper levels. Algorithm 3 describes the details of the discussed method.

Algorithm 3 Intersectional-Coverage

Input: Dataset $\mathcal{D}$, dataset size N, subset size upper bound n, coverage threshold τ, set of attributes x
Output: Coverage for all individual and intersecting groups in x

$\mathcal{L} \leftarrow$LabelSample$(\mathcal{D}, \tau)$
Let $\mathbb{G}$ be the set of all the fully-specified sub-groups at the maximum level
$\mathbb{G}_{agg} \leftarrow$Aggregate$(\mathcal{S}, N, \tau, \mathbb{G}, mult = true)$
$cov \leftarrow$Multiple-Coverage $(\mathcal{D}, N, n, \tau, \mathbb{G}_{agg})$
Let Q = an empty queue
for $\mathbf{g} \in cov$ **do** Q.add$(\langle \mathbf{g}, cvg, cnt \rangle)$
while Q is not empty **do**
    $T \leftarrow Q.del\_top()$
    if $T.cvg$ =true **then**
        for $\forall T.parent$ **do:** Q.add$(\langle T.parent$,true, $T.cnt \rangle)$
    else
        $cnt \leftarrow 0$
        for $\forall P \in T.parent.children - T$ **do**
            $cnt \leftarrow cnt + P.cnt$; $Q.pop(P)$
        if $cnt \geq \tau$ **then** Q.add$(\langle T.parent$,true, $T.cnt \rangle)$
        else Q.add$(\langle T.parent$,false, $T.cnt \rangle)$

5 UTILIZING EXISTING PREDICTORS

In previous sections, we discussed why *solely* employing the standard machine learning algorithms to label the data and coverage identification might be a problematic approach for data coverage, due to reasons such as machine bias and distribution generalization. On the other hand, in presence of accurate and well-developed models, we should be able to utilize them in order to reduce the coverage identification cost – i.e., the number of crowdsourcing tasks. In this section, we adjust our core algorithm for this purpose. In such settings, instead of calling Group-Coverage in subsequent algorithms, one should call our classifier-aware algorithm (Classifier-Coverage – Algorithm 4).

Lack of distribution generalizability is one of the challenges faced when using pre-trained models to label the data [21]. That is, such models may not have the same performance on downstream datasets with different distributions from the data used for training the model. Besides, since our input data is not labeled, we cannot apply techniques such as transfer learning to tune the model for the new setting. As a result, we should utilize pre-trained models as-is which can raise multiple issues. For example, the models could be trained on images with a particular emphasis on facial features, angle, background, etc., and the models may not perform as precisely as reported originally if those aspects are different in our data. To conclude, even if a model has noteworthy performance on some data, there is no guarantee that it will perform as well on another piece of data. We are blind to the features and aspects of the dataset in our problem, hence, there are no concrete methods to choose a well-suited classifier to do the job.

Having said that, applying existing pre-trained classifiers on the dataset, gives us a prediction of what the groups may be. Thus, we define the problem as identifying the group coverage of a labeled dataset. Note that due to the reasons mentioned, we still *need to validate the correctness* of the results obtained by the classifier in order to determine the coverage of a given group. Imagine an example in which a gender classifier applied to a dataset, classifying a set $\mathbf{f}$ as females. In order to identify the coverage for the female group, the main idea is to eliminate the falsely identified females (false positives), namely males, from $\mathbf{f}$. To this end, we apply a similar idea to what we did in Group-Coverage algorithm: we create a crowdsourcing task with all the points in the identified females set $\mathbf{f}$, and ask a *reverse question*: "`Is there any individual in this`

Algorithm 4 Classifier-Coverage

Input: Dataset $\mathcal{D}$, dataset size N, subset size upper bound n, coverage threshold τ, target group $\mathbf{g}$
Output: Coverage of $\mathbf{g}$

Let $\mathcal{G}$ be the set of identified group by the classifier
$\mathcal{G}, \mathcal{L} \leftarrow$ LabelSample $(\mathcal{G}, |\mathcal{G}|, c = 0.1)$
if $\mathcal{L}$.count$(\mathbf{g}') > 0.25|\mathcal{G}|$ **then** $\mathcal{G} \leftarrow$Partition$(\mathcal{D}, \mathcal{G}, n)$
else $\mathcal{G} \leftarrow$Label$(\mathcal{D}, \mathcal{G}, \tau)$
if $|\mathcal{G}| \geq \tau$ **then return** true
else return Group-Coverage $(\mathcal{D} - \mathcal{G}, N, n, \tau - |\mathcal{G}|, \mathbf{g})$

set that is NOT female?". If the answer is yes, it means there are some false positives in this set. Therefore, we take the divide-and-conquer approach by dividing the set into two halves and repeating the question until all false positive instances are eliminated.

A performance issue with this strategy, however, happens when the false positive rate of the classifier for the given group is high. In such settings the divide and conquer strategy keeps dividing the sets into fine granularity, resulting in many small set queries to ask. In such cases, labeling the data points in the female set using point queries to verify the classifier's label might be more efficient.

Following this observation, we propose a sampling phase to estimate the precision of the classifier on the positive group (females in this example). Similar to our proposed method in §4, we choose a small, random sample from the identified females set (in our experiments, we found that a sample size of 10% of the set classified as the given group would be a good choice). In the next step, we ask the crowd to label the sample using point queries and estimate the precision of the classifier, comparing the classifier label and the true label. We experimentally found that if 25% of the sample are false positives, it is safe to say that the precision of the classifier on the positive group is sub-optimal for the group coverage identification task. Based on the estimated precision of the classifier on the target group, our algorithm decides whether to eliminate the false positive objects using either partitioning or labeling strategies.

At the end of this process, we will have a set containing only objects associated with the queried group $\mathbf{g}$ (female in this example). If we already have at least τ instances of a group $\mathbf{g}$, we can determine the coverage and stop the process. However, assuming that the number of discovered instances c' is less than τ, now we have to find at least $\tau - c'$, *false negative* instances of $\mathbf{g}$, i.e., the instances that belong to $\mathbf{g}$ but are classified as not belonging to $\mathbf{g}$ before we can conclude $\mathbf{g}$ is covered (or to verify the number of false positive is less than $\tau - c'$ and hence $\mathbf{g}$ is uncovered). This can be done by applying the algorithm Group-Coverage on the set of objects classified as not $\mathbf{g}$ with the threshold $\tau - c'$. The details of the proposed method can be found in Algorithm 4.

Our experiments in § 6.3.2, utilizing various pre-trained classifiers show that our optimization for coverage identification for classified datasets can achieve remarkable performance.

6 EXPERIMENTS

In this section, we evaluate the performance of our proposed methods for coverage identification of single and multiple demographic groups. Additionally, we explore our heuristic of coverage detection for a single demographic group (gender) on pre-processed image data with pre-trained classifiers. Finally, we deploy our system on a real crowdsourcing platform, Amazon Mechanical Turk, to explore the performance of the discussed methods with real workers.

6.1 Experiment Setup

The experiments were conducted using both synthetic and real datasets. The algorithms were implemented in Python.

- **Synthetic datasets:** To thoroughly assess our algorithms, we create synthetic data with a variety of distributions. As an example, in a single demographic group problem setting for gender, a data point can be either {'F', 'M'}. In these experiments, we simulate the behavior of the crowdworkers in answering queries.
- **Image datasets:** We use image datasets for the purpose of experiments on MTurk and applying pre-trained classifiers on the data. In this section, we used slices of FERET DB [28] and UTKFace [42]. *FERET DB* is a dataset of 14,126 images of 1199 individuals taken from a variety of angles. *UTKFace* is also an image dataset consisting of over 20,000 annotated facial images.

Evaluation Plan: We evaluate the performance of Group-Coverage and the optimizations for multiple non-intersectional and intersectional groups as well as classified data using pre-trained classifiers. We report the number of tasks required for each experiment setting.

We use a straightforward procedure to identify group coverage as a baseline method to compare our algorithm's results to. In this method, each task is created containing only one single data point and is asked about. There are two outcomes for this algorithm which are represented in Algorithm 7 (Base-Coverage)[4]: Either at some point, τ instances of objects associated with the group are identified and hence, the group is covered, or the algorithm goes through all the data and determines that the group is uncovered.

Default Values: To evaluate the performance of our algorithms, we fix the value of some of the introduced parameters in §3-5. The default value of c is 2 and we fix τ and n as 50 in all experiments except their respective parameter analysis experiments.

6.2 Summary of Results

Our proposed Group-Coverage algorithm achieved remarkable results in group coverage identification in our experiments. Even at the worst possible case (where the number of instances associated with the group in the dataset is close to the coverage threshold), both the synthetic and MTurk experiments with the real crowdworkers showed that our algorithm needs a significantly small number of tasks compared to the size of the dataset in order to achieve results. We also show that the upper bound discussed in 3.2 is in fact tight.

Our optimizations on multiple groups cases (both intersectional and non-intersectional) proved to be effective in most cases compared to a brute force approach utilizing the Group-Coverage algorithm to identify the coverage of multiple groups. Additionally, the optimization of classified data using pre-trained classifiers achieved notable results in most cases, decreasing the number of required tasks by approximately 80% in some.

6.3 Proof of Concept

6.3.1 Amazon Mechanical Turk. In this experiment, we evaluate our proposed method and the crowd's performance on a live crowdsourcing platform. We defined a female coverage identification task and published our HITs on Amazon Mechanical Turk. Each HIT

[4]Due to the space limitations, the algorithm has been provided in the technical report [6].

dataset	classifier	accuracy	precision on female group	false positive elimination strategy	#HITs	Group-Coverage
FERET DB (females=403, males=591)	DeepFace (opencv)	79.57	99.5	Partition	14	80
	DeepFace (Retinaface)	84.1	100.0	Partition	17	80
	BaseCNN	64.48	59.19	Label	84	80
UTKFace (females=200, males=2800)	DeepFace (opencv)	93.56	52.02	Label	97	51
	DeepFace (retinaface)	94.16	56.15	Label	89	51
	BaseCNN	97.6	74.8	Label	69	51
UTKFace (females=20, males=2980)	DeepFace (opencv)	96.53	8.0	Lable	134	221
	DeepFace (retinaface)	96.43	10.09	Lable	143	221
	BaseCNN	97.6	21.59	Lable	122	221

Table 1: The results of female group coverage detection on gender classified datasets

Dataset	#HITs
FERET DB-1 (females=215, males=1307)	74
FERET DB-2 (females=403, males=591)	80

Table 2: Coverage identification for female on Amazon MTurk

contained a set of initial $n = 50$ images to present a reasonable workload to the crowd, with a maximum assignment to 3 workers for quality control. The layout of the HITs was designed as Figure 2. The crowd was asked to answer the questions with yes or no.

For the truth aggregation, we employed the off-the-shelf Majority Vote [43] technique, in which we assigned the same HIT to 3 workers and took the majority vote as the truth. One interesting observation that was made is that despite the relatively low number of assignments per HIT, there was no error made in any of the majority votes received from the crowd, which further supports our idea about the tasks being relatively easy and straight-forward for the crowd and fault-proof to an extent.

We employed the fixed price model as our pricing strategy. In our first set of experiments, each HIT price was set to \$0.1. In the next experiment, we decreased the reward for each HIT to \$0.05. Surprisingly, this did not discourage the workers to accept and complete our tasks. Overall, we paid a total of \$35.1 to the workers and \$7.02 to Amazon MTurk as service charges.

We used two different subsets of the FERET dataset. With $\tau = 50$, the results from each experiment setting can be found in Table 2 and Figure 5i with a comparison to the baseline PointCoverage method and the theoretical upper bound on the maximum number of tasks ($\frac{N}{n} + \tau \log(n)$).

6.3.2 Existing predictors. To investigate the performance of the existing predictors on image datasets and our strategy to detect the coverage of a dataset utilizing these models, we ran a number of experiments using DeepFace [32] and another CNN-based facial demographic classifier [19] to predict the gender of the individual in datasets. We used a subset of images of unique individuals from the FERET dataset [28], and two 3,000-point subsets of UTKFace [42] with different distribution of females to evaluate the results for both covered and uncovered cases. We applied DeepFace with opencv and retinaface as the underlying face detectors. Next, we gave the predicted labels and the datasets to our algorithm. The results are reported in Table 1.

For each experiment, the accuracy and the precision of the classifier on the dataset are calculated separately. As discussed before, the accuracy of some classifiers can vary by a large margin on different data. Moreover, a high level of accuracy does not necessarily guarantee reasonable precision in predicting the class of data points. More specifically, both of the classifiers had a relatively high accuracy in the classification task on the UTKFace dataset, but both also had a low precision in their prediction for the female demographic group, which further proves the fact that the performance of the existing predictor for sensitive groups is questionable in many cases.

In addition, the results show that our heuristics make the right decision for which technique to use in most cases, leading to significantly fewer necessary tasks to get the result. Compared to our Group-Coverage algorithm used standalone, the proposed techniques can produce significantly better results in most cases and still competitive results in others.

6.4 Performance Evaluation

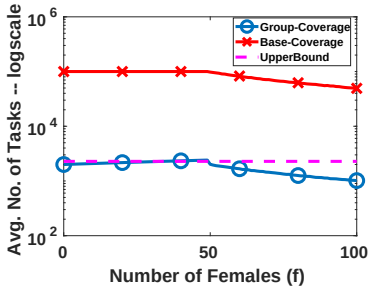
In the following sections, we present the results of our experiments using the stated settings. First, we evaluate the performance of the Group-Coverage algorithm with varying parameters τ, n, N in §6.4.1. Next, we evaluate the optimizations for multiple non-intersectional and intersectional groups in §6.4.2.

6.4.1 Group-Coverage Algorithm. In the following, we evaluate the performance of our proposed Group-Coverage algorithm. To this end, we designed a simulation to reflect the procedure that the crowd would be presented with to carry out the tasks. The objective of the simulation is to first identify whether the dataset is covered with respect to a demographic group and determine the total number of tasks required to identify the coverage of a given group.

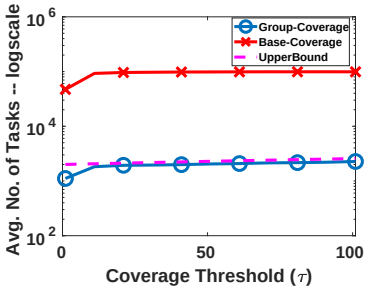
For this purpose, assuming we are interested in identifying the coverage of female, we generate a dataset containing males and females and shuffle it randomly to prepare for the experiment. Each experiment with particular variables is run multiple times to better capture the effect of the dataset's underlying distribution on the results. In these sets of experiments, we study the impact of the scope of parameters on the end results.

Varying τ. First, we analyze the relationship between the coverage threshold and the number of females in the dataset and its impact on the number of necessary tasks to get the results. Figure 5a illustrates the number of required tasks when there exist $[0, 2\tau]$ items of the demographic group in the dataset. We have a dataset of size 100K and we select the coverage threshold as 50. It can be observed that the largest number of queries is needed when the number of females (f) is close to τ. Conversely, the farther f gets from τ, the fewer tasks are required to get to a conclusion. This observation is consistent with the discussion in §3.2; with too few or too large quantities of f in the dataset, our algorithm's results appear to be further from the upper bound.

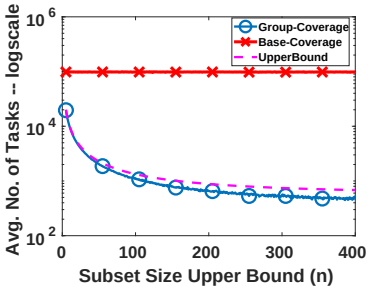
Figure 5b shows the results of running the algorithm with different coverage thresholds. The coverage threshold is varied from 0.001% to 0.1% of the dataset size and there are exactly τ females



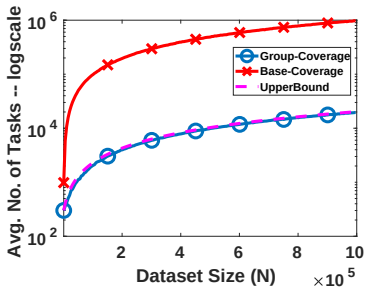
(a) The relation between #females in the dataset, coverage threshold, and #questions



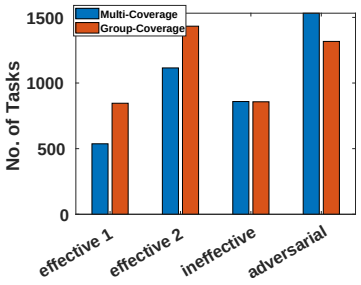
(b) Varying coverage threshold



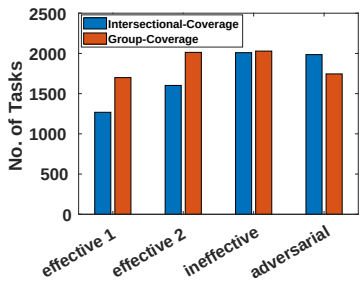
(c) Varying subset size



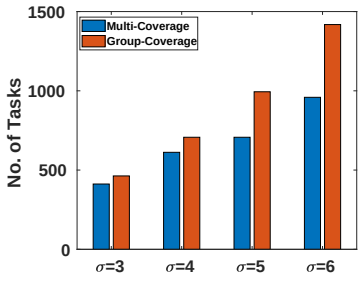
(d) Varying dataset size



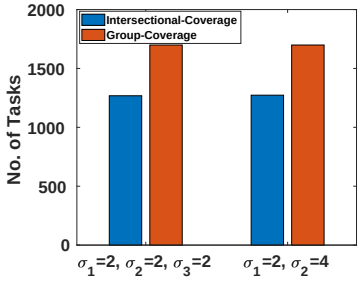
(e) Multiple non-intersectional groups optimization ($\sigma = 4$) vs. Group-Coverage



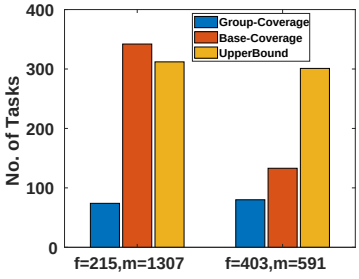
(f) Intersectional groups optimization ($\sigma_1 = 2, \sigma_2 = 2, \sigma_3 = 2$) vs. Group-Coverage



(g) Multiple groups in one attribute with $\sigma = 3, 4, 5, 6$



(h) Multiple groups in two attributes with $\sigma_1 = 2, \sigma_2 = 4$ and $\sigma_1 = 2, \sigma_2 = 2, \sigma_3 = 2$



(i) Group-Coverage algorithm on Amazon Mechanical Turk

Figure 5: Performance evaluation for Group-Coverage, Multiple-Coverage, and Intersectional-Coverage algorithms

at each run. Naturally, when the coverage threshold increases, the algorithm needs to cover more grounds to produce results. This also shows that the relationship between the coverage threshold and the cost is linear as discussed in §3.2. Note that the results in this figure demonstrate the case where $f = \tau$, which is the situation that requires the maximum number of tasks to get to a decision and is very close to the theoretical upper bound.

Varying n. This experiment is designed to study the impact of the subsets size upper bound on the algorithm's outcome. We set the coverage threshold to 50 in a dataset of size 100,000 while maintaining 50 instances of females in the dataset. In Figure 5c, we can see a substantial jump in the number of tasks when the subset size increases from around 10 to 20. Moreover, the result does not change significantly after that even with a notable increase in the subsets size. This confirms the logarithmic nature of the subsets size upper bound parameter in the algorithm.

When determining the set size, one must take the ability of the crowd to identify the subject of the task at hand into consideration. While selecting larger n might lead to fewer required tasks, it is likely that we obtain less reliable answers from the crowd due to the large number of items presented all at once. Additionally, increasing the initial subset size will not significantly impact on the end results.

Varying N. To assess the performance of our algorithm in a variety of datasets, we ran experiments for datasets of size 1K to 1M.

Figure 5d illustrates the results of the algorithm for varying dataset sizes. As expected, the number of required tasks to determine the results grows linearly with the size of the dataset, but never exceeds 6%. In other words, our results show that in practice, we can determine the group coverage for a dataset with tasks no more than 6% of the dataset size at the very worst case.

Setting	Description
effective 1	3 uncovered minorities; their aggregated super-group is uncovered
effective 2	3 covered minorities
ineffective	2 uncovered and one covered minority
adversarial	3 uncovered minorities; their aggregated super-group is covered

Table 3: Experiment settings for multiple groups

6.4.2 Optimizations for multiple groups. To evaluate the performance of our proposed method in identifying the coverage for multiple groups, we take a similar approach as the previous section. We create a synthetic dataset comprising of data points that can correspond to $\sigma = 3, 4, 5, 6$ distinct demographic groups for the non-intersectional case, and two datasets, one with 2 attributes with cardinalities $\sigma_1 = 2, \sigma_2 = 4$ and the other with 3 binary attributes ($\sigma_1 = 2, \sigma_2 = 2, \sigma_3 = 2$) for the intersectional case. In a dataset of 10K points, with a threshold of 50 and a subset size of 50, we vary the number of items for each group in the dataset to simulate different combinations of settings and run the algorithm for each variation. Additionally, we run the Group-Coverage algorithm for each group independently to compare our results. In our experiments, we found out that while our heuristics on multiple groups can perform very well in some cases, it can also appear to be ineffective or worse than the brute force in some other. The results of the experiments for these cases are shown in Figures 5e and 5f.

Each of the bars defined as effective 1, effective 2, ... represent a different setting with respect to the number of instances associated with each group which is further described in Table 3.

The adversarial case, in which there are multiple uncovered groups with summation of items greater than τ in the dataset, it is likely that our heuristic fails in aggregating these groups into a super-group since the probability of having instances of these groups in the sample is significantly low. Thus, the super-group is covered and the algorithm needs to run for each of the subgroups individually. This imposes a penalty on the total number of tasks and makes it an adversarial case for our heuristic. To conclude, we can expect that our method works very well or with little difference compared to brute force in some cases while failing in others.

Figure 5g shows the results for the Multiple-Coverage algorithm for attributes with various cardinalities. Considering cases where our heuristic is effective, as the cardinality of the attribute increases, the total required tasks in Multiple-Coverage grows more slowly than the brute force, resulting in a larger gap between the two methods as the cardinality increases.

Figure 5h represents the results of the Intersectional-Coverage algorithm for two cases, one with 2 attributes with cardinalities $\sigma_1 = 2, \sigma_2 = 4$ and the other with 3 binary attributes ($\sigma_1 = 2, \sigma_2 = 2, \sigma_3 = 2$). As expected, with the same settings, the results for each of these cases are similar, with the number of fully-specified subgroups at the maximum level for both cases being equal. In other words, in the case of intersectional groups with multiple attributes, the only important feature is the cardinality of the attributes rather than the number of attributes.

7 RELATED WORKS

Crowd-sourcing for Bias Detection. [15] proposes a crowd-sourcing workflow to facilitate sampling bias discovery in visual datasets with the help of human-in-the-loop. This workflow takes a visual data set as an input and outputs a list of potential biases of the data set. There are three steps in this workflow. The first step is *Question Generation* and the crowd inspects random samples of images from the input dataset and describes their similarity using a question-answer pair. The next step is *Answer Collection* in which the crowd review separates random samples of images from the input dataset and provides answers to questions solicited from the previous step. Finally, in the third step called *Bias Judgement* the crowd judge whether statements of the visual dataset that are automatically generated using the questions and answers collected accurately reflect the real world. While this method might prove effective in detecting some bias in small and simple visual sets, it relies a great deal on the crowd's sense of identifying any possible bias in the data presented to them, which can lead to unreliable results as a consequence of different individuals' perception of bias. Additionally, the method seems to address the problem in an ad-hoc manner and there is little control over the domain of the results obtained from each step.

Set queries. [30] first introduced the idea of filtering a set of data based on a particular property using humans. For the purpose of top-k and group-by queries, [7] uses the crowd to answer *type* set question which has "yes" or "no" answer based on whether the data points in a set have the same type, which is similar to our notion of set queries on a target demographic group.

Coverage. The notion of data *coverage* has been studied across different settings [1–4, 18, 22, 24, 33, 37]. For categorical data, uncovered regions are identified in form of value combinations (e.g. Hispanic Females) called patterns. A pattern is uncovered if there are not enough samples matching it [3, 18, 22]. *Coverage* on continuous space is studied in [4]. Accordingly, lack of *coverage* is identified as any point in the data space that doesn't have enough points in a fixed-radius neighborhood around it. Existing works in coverage detection have so far only focused on tabular data and, to the best of our knowledge, our paper is the first to propose techniques for identifying sufficient coverage in image datasets.

8 CONCLUSION

In this paper, we studied the problem of coverage identification in image data. This problem is motivated by the historical representation bias in various forms of data, and specifically the inefficiency of the existing supervised or unsupervised learning methods in performing equally well for minority groups on image data. We proposed an efficient algorithm to identify the coverage of a demographic group across the dataset and showed that the number of required tasks is optimal and close to the theoretical lower bound, and introduced practical heuristics to expand our solution for multiple non-intersectional or intersectional groups. We also presented an optimization method for detecting group coverage in datasets labeled by the existing, pre-trained predictors.

In this work, we focused on image data as a specific form of multimedia data. We hope to find equally efficient methods to identify data coverage in other forms of multimedia data such as video in our future work. In addition, our goal in this paper was mainly to minimize the cost of crowdsourcing by minimizing the total number of required tasks. We consider extending our techniques to support various pricing models as part of our future work.

REFERENCES

[1] Chiara Accinelli, Barbara Catania, Giovanna Guerrini, and Simone Minisi. 2021. The impact of rewriting on coverage constraint satisfaction.. In *EDBT/ICDT Workshops*.

[2] Chiara Accinelli, Simone Minisi, and Barbara Catania. 2020. Coverage-based Rewriting for Data Preparation.. In *EDBT/ICDT Workshops*.

[3] Abolfazl Asudeh, Zhongjun Jin, and HV Jagadish. 2019. Assessing and remedying coverage for a given dataset. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 554–565.

[4] Abolfazl Asudeh, Nima Shahbazi, Zhongjun Jin, and H. V. Jagadish. 2021. Identifying Insufficient Data Coverage for Ordinal Continuous-Valued Attributes. In *SIGMOD*. ACM.

[5] Abolfazl Asudeh, Gensheng Zhang, Naeemul Hassan, Chengkai Li, and Gergely V. Zaruba. 2015. Crowdsourcing Pareto-Optimal Object Finding By Pairwise Comparisons. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (Melbourne, Australia) *(CIKM '15)*. Association for Computing Machinery, New York, NY, USA, 753–762. https://doi.org/10.1145/2806416.2806451

[6] Anonymouse Author(s). 2022. Details Omitted Due to Double-blind Policy. Anonymized Link to the Appendix (for review): https://anonymous.4open.science/r/ImageDataCvgCrwd-DE60/techReport.pdf.

[7] Rubi Boim, Ohad Greenshpan, Tova Milo, Slava Novgorodov, Neoklis Polyzotis, and Wang-Chiew Tan. 2012. Asking the Right Questions in Crowd Data Sourcing. In *2012 IEEE 28th International Conference on Data Engineering*. 1261–1264. https://doi.org/10.1109/ICDE.2012.122

[8] Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*. PMLR, 77–91.

[9] L Elisa Celis and Vijay Keswani. 2020. Implicit diversity in image summarization. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW2 (2020), 1–28.

[10] Kate Crawford. 2013. The hidden biases in big data. *Harvard business review* 1, 4 (2013).

[11] Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 28, 1 (1979), 20–28.

[12] Nicholas Diakopoulos. 2015. Algorithmic accountability: Journalistic investigation of computational power structures. *Digital journalism* 3, 3 (2015), 398–415.

[13] John R Feiner, John W Severinghaus, and Philip E Bickler. 2007. Dark skin decreases the accuracy of pulse oximeters at low oxygen saturation: the effects of oximeter probe type and gender. *Anesthesia & Analgesia* 105, 6 (2007), S18–S23.

[14] Batya Friedman and Helen Nissenbaum. 1996. Bias in computer systems. *ACM Transactions on Information Systems (TOIS)* 14, 3 (1996), 330–347.

[15] Xiao Hu, Haobo Wang, Anirudh Vegesana, Somesh Dube, Kaiwen Yu, Gore Kao, Shuo-Han Chen, Yung-Hsiang Lu, George K Thiruvathukal, and Ming Yin. 2020. Crowdsourcing Detection of Sampling Biases in Image Datasets. In *Proceedings of The Web Conference 2020*. 2955–2961.

[16] Basileal Imana, Aleksandra Korolova, and John Heidemann. 2021. Auditing for discrimination in algorithms delivering job ads. In *Proceedings of the Web Conference 2021*. 3767–3778.

[17] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*. 64–67.

[18] Zhongjun Jin, Mengjing Xu, Chenkai Sun, Abolfazl Asudeh, and HV Jagadish. 2020. MithraCoverage: A System for Investigating Population Bias for Intersectional Fairness. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2721–2724.

[19] Shreyansh Joshi. 2020. Age and Gender Prediction from Facial Images. https://github.com/ShreyanshJoshi/Facial-Demographics-using-CNN.

[20] Matthew Kay, Cynthia Matuszek, and Sean A Munson. 2015. Unequal representation and gender stereotypes in image search results for occupations. In *Proceedings of the 33rd annual acm conference on human factors in computing systems*. 3819–3828.

[21] Bogdan Kulynych, Yao-Yuan Yang, Yaodong Yu, Jarosław Błasiok, and Preetum Nakkiran. 2022. What You See is What You Get: Distributional Generalization for Algorithm Design in Deep Learning. *arXiv preprint arXiv:2204.03230* (2022).

[22] Yin Lin, Yifan Guan, Abolfazl Asudeh, and HV Jagadish. 2020. Identifying insufficient data coverage in databases with multiple relations. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2229–2242.

[23] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–35.

[24] Yuval Moskovitch and HV Jagadish. 2020. Countata: dataset labeling using pattern counts. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2829–2832.

[25] Yuval Moskovitch and H. V. Jagadish. 2020. COUNTATA: Dataset Labeling Using Pattern Counts. *PVLDB* 13, 12 (2020), 2829–2832.

[26] Rajeev Motwani and Prabhakar Raghavan. 1995. *Randomized algorithms*. Cambridge university press.

[27] M. Mulshine. 2015. A major flaw in Google's algorithm allegedly tagged two black people's faces with the word 'gorillas'. Business Insider.

[28] National Institute of Standards and Technology. 2011. Face Recognition Technology Database (FERET DB). https://www.nist.gov/itl/products-and-services/color-feret-database.

[29] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Emre Kıcıman. 2019. Social data: Biases, methodological pitfalls, and ethical boundaries. *Frontiers in Big Data* 2 (2019), 13.

[30] Aditya G. Parameswaran, Hector Garcia-Molina, Hyunjung Park, Neoklis Polyzotis, Aditya Ramesh, and Jennifer Widom. 2012. CrowdScreen: Algorithms for Filtering Data with Humans. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (Scottsdale, Arizona, USA) *(SIGMOD '12)*. Association for Computing Machinery, New York, NY, USA, 361–372. https://doi.org/10.1145/2213836.2213878

[31] Adam Rose. 2010. Are Face-Detection Cameras Racist? Time Business.

[32] Sefik Ilkin Serengil and Alper Ozpinar. 2020. LightFace: A Hybrid Deep Face Recognition Framework. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE, 23–27. https://doi.org/10.1109/ASYU50717.2020.9259802

[33] Nima Shahbazi, Yin Lin, Abolfazl Asudeh, and H. V. Jagadish. 2022. A Survey on Techniques for Identifying and Resolving Representation Bias in Data. (2022). https://doi.org/10.48550/arxiv.2203.11852

[34] Yaron Singer and Manas Mittal. 2013. Pricing Mechanisms for Crowdsourcing Markets. In *Proceedings of the 22nd International Conference on World Wide Web* (Rio de Janeiro, Brazil) *(WWW '13)*. Association for Computing Machinery, New York, NY, USA, 1157–1166. https://doi.org/10.1145/2488388.2488489

[35] Adish Singla and Andreas Krause. 2013. Truthful Incentives in Crowdsourcing Tasks Using Regret Minimization Mechanisms. In *Proceedings of the 22nd International Conference on World Wide Web* (Rio de Janeiro, Brazil) *(WWW '13)*. Association for Computing Machinery, New York, NY, USA, 1167–1178. https://doi.org/10.1145/2488388.2488490

[36] Harini Suresh and John Guttag. 2021. A framework for understanding sources of harm throughout the machine learning life cycle. In *Equity and Access in Algorithms, Mechanisms, and Optimization*. 1–9.

[37] Ki Hyun Tae and Steven Euijong Whang. 2021. Slice tuner: A selective data acquisition framework for accurate and fair machine learning models. In *Proceedings of the 2021 International Conference on Management of Data*. 1771–1783.

[38] Antonio Torralba and Alexei A Efros. 2011. Unbiased look at dataset bias. In *CVPR 2011*. IEEE, 1521–1528.

[39] Jing Wang, Panagiotis G Ipeirotis, and Foster Provost. 2011. Managing crowdsourcing workers. In *The 2011 winter conference on business intelligence*. Citeseer, 10–12.

[40] Peter Welinder, Steve Branson, Pietro Perona, and Serge Belongie. 2010. The multidimensional wisdom of crowds. *Advances in neural information processing systems* 23 (2010).

[41] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier Movellan, and Paul Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in neural information processing systems* 22 (2009).

[42] Zhifei Zhang, Yang Song, and Hairong Qi. 2017. Age Progression/Regression by Conditional Adversarial Autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

[43] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. 2017. Truth Inference in Crowdsourcing: Is the Problem Solved? *Proc. VLDB Endow.* 10, 5 (jan 2017), 541–552. https://doi.org/10.14778/3055540.3055547

APPENDIX

A PROOFS

Lemma 1 *(Correctness) The* Group-Coverage *algorithm successfully identifies if a group* $\mathbf{g}$ *is covered or not, i.e., if there are at least* τ *instances of* $\mathbf{g}$ *in the data set* $\mathcal{D}$.

Proof. Each set query with a yes answer contains at least one object belonging to the group $\mathbf{g}$. Using this, the algorithm maintains a lower bound cnt on the $|\mathbf{g}|$ in $\mathcal{D}$. That is, $cnt \leq |\mathbf{g}|$. The algorithm returns `true` when $cnt = \tau$. When $cnt = \tau$, $\mathbf{g}$ is covered, because $\tau = cnt \leq |\mathbf{g}|$. The algorithm returns false when the queue is empty and $cnt < \tau$. For sets with yes answers, the algorithm divides the set into two halves, unless the set size is 1. As a result, when the queue is empty, all the set questions with yes answers have had size 1, otherwise, the queue would have not been empty. Therefore, $|\mathbf{g}| = cnt < \tau$, meaning that $\mathbf{g}$ is uncovered. □

B PSEUDO-CODES

Algorithm 5 Partition & Label

function Partition($\mathcal{D}, \mathcal{G}, n$)
  Let Q = an empty queue
  for $i \leftarrow 0$ to N with step size n **do**:
    $t \leftarrow \{t_i, \cdots, t_j\}$
    $Q.add(t)$
  Let S = an empty set
  while Q is not empty **do**
    $T \leftarrow Q.del\_top()$
    $(i, j) \leftarrow (T.b\_index, T.e\_index)$
    ans ←AskQuestion($\{t_i, \cdots, t_j\}, \mathbf{g}'$)
    if ans=no **then**
      S.add($\{t_i, \cdots, t_j\}$)
    else
      if $j > i$ `/*if setsize>1*/` **then**
        $T_1 \leftarrow \{t_i, t_{\lfloor \frac{i+j}{2} \rfloor}\}$
        $T_2 \leftarrow (t_{\lfloor \frac{i+j}{2} \rfloor + 1}, t_j)$
        $Q.add(T_1)$; $Q.add(T_2)$
  return S
function Label($\mathcal{D}, \mathcal{G}, \tau$)
  $cnt \leftarrow 0$
  for $t \in \mathcal{G}$ **do**
    l ←PointQuery(t)
    if $l \neq \mathbf{g}$ **then** $\mathcal{G}$.Remove(t)
    else $cnt \leftarrow cnt + 1$
    if $cnt \geq \tau$ **then**
      break
  return $\mathcal{G}$

Algorithm 6 Label Samples & Aggregate

function LabelSamples($\mathcal{D}, \tau, c = 2$)
  for $c\tau$ random samples t from $\mathcal{D}$ **do**
    $l \leftarrow$ PointQuery(t)
    $\mathcal{L}.add(\langle t, l \rangle)$; $\mathcal{D}.remove(t)$
  return $\mathcal{D}$, $\mathcal{L}$
function Aggregate($\mathcal{L}, N, \tau, \mathbb{G}, mult = false$)
  sort $\mathbb{G}$ based on $\mathcal{L}$.count($\mathbf{g}$), $\mathbf{g} \in \mathbb{G}$, ascending
  $sum \leftarrow 0$; $\mathcal{G} \leftarrow \{\}$
  for $\mathbf{g} \in \mathbb{G}$ **do**
    $E_\mathbf{g} \leftarrow \frac{\mathcal{L}.\text{count}(\mathbf{g})}{|\mathcal{L}|} \times N$
    if: $mult$ = true **then** $\mathcal{G} \leftarrow \mathcal{G} \mid \mathcal{G}$.parent= $\mathbf{g}$.parent
    if:$sum + E_\mathbf{g} < \tau$ **then** $\mathcal{G}$.add($\mathbf{g}$); $sum \leftarrow sum + E_\mathbf{g}$
    else: $\mathbb{G}_{agg}$.add($\mathcal{G}$) ; $\mathcal{G} \leftarrow \{\mathbf{g}\}$; $sum \leftarrow E_\mathbf{g}$
  return $\mathbb{G}_{agg}$.add($\mathcal{G}$)

Algorithm 7 Base-Coverage($\mathcal{D}, \tau, \mathbf{g}$)

Input: Dataset $\mathcal{D}$, coverage threshold τ, and target group $\mathbf{g}$
Output: Coverage of group $\mathbf{g}$

$cnt \leftarrow 0$
for $\forall t \in \mathcal{D}$ **do**
  ans ←AskQuestion($t, \mathbf{g}$)
  if ans =**true then** $cnt \leftarrow cnt + 1$
  if $cnt = \tau$ **then return true** `// covered`
return false `//uncovered`