Reef: Automating Weak Supervision to Label Training Data

Paroma Varma
Stanford University
paroma@stanford.edu

Christopher Ré
Stanford University
chrismre@cs.stanford.edu

ABSTRACT

As deep learning models are applied to increasingly diverse and complex problems, a key bottleneck is gathering enough high-quality training labels tailored to each task. Users therefore turn to *weak supervision*, relying on imperfect sources of labels like user-defined heuristics and pattern matching. Unfortunately, with weak supervision users have to design different labeling sources *for each task*. This process can be both time consuming and expensive: domain experts often perform repetitive steps like guessing optimal numerical thresholds and designing informative text patterns. To address these challenges, we present Reef, the first automated weak supervision system that generates heuristics based on a small, labeled dataset to assign training labels to a larger, unlabeled dataset. Reef generates heuristics that each labels only the subset of the data it is accurate for, and iteratively repeats this process until the heuristics together label a large portion of the unlabeled data. We also develop a statistical measure that guarantees the iterative process will automatically terminate before it degrades training label quality. Compared to the best known user-defined heuristics developed over several days, Reef automatically generates heuristics in under five minutes and performs up to 9.74 F1 points better. In collaborations with users at several large corporations, research labs, Stanford Hospital and Clinics, and on open source text and image datasets, Reef outperforms other automated approaches like semi-supervised learning by up to 14.35 F1 points.

1. INTRODUCTION

The success of machine learning for tasks like image recognition and natural language processing [11, 13] has has ignited interest in using similar techniques for a variety of tasks. However, gathering enough training labels is a major bottleneck in applying machine learning to new tasks. In response, there has been a shift towards relying on *weak supervision*, or methods that can assign noisy traning labels to unlabeled data, like crowdsourcing [8, 21, 55], knowledge bases in distant supervision [7, 29], and user-defined heuristics [35, 36, 47]. Over the past few years, we have been part of the broader effort to enhance methods based on user-defined heuristics to extend their applicability to text, image, and video data for tasks in computer vision, medical imaging, bioinformatics and knowledge base construction [36, 3, 47].

Through our engagements with users at large companies, we find that experts spend a significant amount of time *designing* these weak supervision sources. As deep learning techniques are adopted for unconventional tasks like analyzing codebases and now commodity tasks like driving marketing campaigns, the few domain experts with required knowledge to write heuristics cannot reasonably keep up with the demand for several specialized, labeled training datasets. Even machine learning experts, such as researchers at the computer vision lab at Stanford, are impeded by the need to crowdsource labels for the relevant task before even starting to build models for novel visual prediction tasks [23, 22]. This raises an important question: *can we make weak supervision techniques easier to adopt by automating the process of generating heuristics that assign training labels to unlabeled data?*

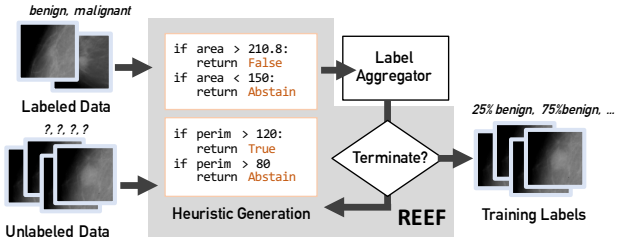

Figure 1: Reef uses a small labeled and a large unlabeled dataset to iteratively generate heuristics and uses existing label aggregators to output training labels for the unlabeled dataset.

The key challenge in automating weak supervision lies in replacing the human reasoning that drives heuristic development. However, in our collaborations with users with varying levels of machine learning expertise, we noticed that the process to develop these weak supervision sources can be fairly repetitive. For example, radiologists at the Stanford Hospital and Clinics have to guess the correct threshold for each heuristic that uses a geometric property of a tumor to determine if it is malignant (example shown in Figure 1). We instead take advantage of *a small, labeled dataset to automatically generate noisy heuristics*. While this labeled dataset is too small to train an end model, using it to automatically assign training labels to a large amount of unlabeled data can improve over training with only the labeled set by 12.12 F1 points.

To aggregate labels from these heuristics, we can improve over majority vote by relying on existing statistical techniques in weak supervision that can model the noise in and correlation among these heuristics [36, 3, 47, 1, 38, 45]. However, these techniques were intended to work with user-designed labeling sources and therefore have limits on how robust they are. Automatically generated heuristics can be much noisier than the what these models are robust to and introduce the following challenges:

Accuracy. In our experience, users develop heuristics that assign highly accurate labels to a *subset* of the unlabeled data. An automated method therefore has to properly model

this trade-off between accuracy and coverage (how many datapoints it labels) for each heuristic based only on the small, labeled dataset. Empirically, we find that generating heuristics that *each* label all the datapoints can degrade end model performance by up to 20.69 F1 points.

Diversity. Since each heuristic has limited coverage, users develop multiple heuristics that each label different subsets of the unlabeled data to ensure a large portion of the unlabeled data receives a label. In an automated approach, we could mimic this by maximizing the number of unlabeled datapoints the heuristics label as a set. However, this approach can select heuristics that cover a large portion of the data but have poor performance; therefore, there is a need to account for both the diversity and performance of the heuristics *as a set*. Empirically, this improves end model performance by up to 18.20 F1 points compared to simply selecting the heuristic set that labels the maximum number of datapoints.

Termination Condition. Users stop generating heuristics when they have exhausted their domain knowledge while an automated method lacks the ability to recognize when the set of heuristics it has generated is sufficient. It can therefore continue to generate heuristics that deteriorate the overall quality of the training labels assigned to the unlabeled data, such as heuristics that are worse than random for the unlabeled data. Generating as many heuristics as possible without accounting for performance on the unlabeled dataset can affect end model performance by up to 7.09 F1 points.

Our Approach. To address the challenges above, we introduce Reef, an automated system that takes as input a small labeled dataset and a large unlabeled dataset and outputs probabilistic training labels for the unlabeled data, as shown in Figure 1. These training labels can be used to train a downstream machine learning model of choice. Over the past 9 months, our experience working with Reef users from research labs, hospitals and major chip manufacturing and social network companies helped us design Reef such that it outperforms user-defined heuristics and crowdsourced labels by up to 9.74 F1 points and 13.80 F1 points in terms of end model performance, respectively.

Reef is iterative in nature, appending a new heuristic to the set that will assign labels to the unlabeled dataset at each iteration. Each iteration consists of the data flowing through the following components of the Reef architecture:

Synthesizer for Accuracy. To account for the trade-off between the accuracy and coverage of each heuristic, the *synthesizer* (Section 3.1) generates heuristics based on the labeled set and adjusts its labeling pattern to only assign labels if the heuristic has high confidence and abstain otherwise. The synthesizer relies on a small number of different primitives, or features of the data to generate several simple models like decision trees, improving over using a single classifier by 12.12 F1 points in terms of end model performance. These primitives are user-defined and part of open source libraries [32, 46] and data models in existing weak supervision frameworks [35, 54]. We utilize several forms of simple primitives like the bag-of-words representation and bounding box attributes in our evaluation.

Pruner for Diversity To ensure that the set of heuristics is diverse and assigns high-quality labels to large portion of the unlabeled data, the *pruner* (Section 3.2) ranks the heuristics the synthesizer generates by both their performance on the labeled set and coverage on the unlabeled set. It selects the best heuristic at each iteration and adds it to the collection of existing heuristics. In terms of end model performance, this method performs up to 6.57 F1 points better than relying only on performance to rank heuristics.

Verifier to Determine Termination Condition The verifier uses existing statistical techniques to aggregate labels from the heuristics into probabilistic labels for the unlabeled datapoints [47, 36, 3]. However, the automated heuristic generation process can surpass the noise these techniques are robust to and can therefore degrade end model performance by up to 7.09 F1 points. To address this, we develop a statistical measure that uses the small, labeled set to determine whether the noise in the generated heuristics is below the threshold these techniques can handle (Section 4).

Contribution Summary. This paper describes Reef, the first automated weak supervision system to use a small labeled dataset to generate heuristics that can assign training labels to a large, unlabeled dataset. A summary of our contributions and outline are as follows:

- We describe the system architecture, the iterative process of generating heuristics, and the optimizers used in the synthesizer, pruner and verifier in Section 3. We also show that our automated optimizers can affect end model performance by up to 20.69 F1 points in Section 5.
- We present a theoretical guarantee that Reef will terminate the iterative process *before* the noise in heuristics surpasses the noise threshold statistical techniques are robust to in Section 4. We show how this theoretical result translates to improving end model performance by up to 7.09 F1 points compared to generating as many heuristics as possible in Section 5.
- We evaluate our system in Section 5 by training downstream models using labels Reef generates. We report on collaborations with Stanford Hospital and Clinics and Stanford Computer Vision Lab, and on open source datasets representative of collaborations with industry, analyzing structured and unstructured data like text and images. We show that heuristics generated in as little as five minutes can improve over hand-crafted heuristics developed over several days by up to 9.74 F1 points. We also compare to automated methods like semi-supervised learning, which Reef outperforms by up to 14.35 F1 points.

2. SYSTEM OVERVIEW

We first describe the input and output for Reef and introduce notation used in the rest of paper. We then briefly describe statistical techniques Reef relies on to learn heuristic accuracies and provide an overview of the Reef pipeline. Finally, we describe in further detail the challenges associated with designing this pipeline.

2.1 Input and Output Data

Input Data. The input to Reef is a labeled dataset O_L with N_L datapoints and an unlabeled dataset O_U with N_U datapoints. Each datapoint is defined by its associated *primitives*, or characteristics of the data, and a label. Therefore,

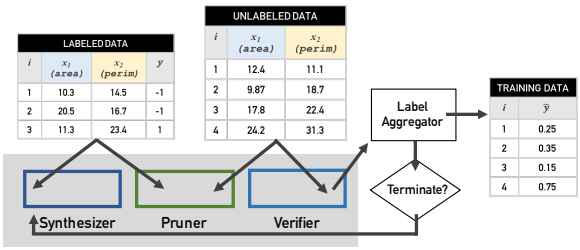

Figure 2: Reef input and output. Users input a labeled and unlabeled dataset and Reef outputs training labels for the unlabeled dataset.

the inputs to the system can be represented as

$$\{x_i, y_i^*\}_{i=1}^{N_L}, \text{ (for the labeled set } O_L\text{), and}$$

$$\{x_i\}_{i=1}^{N_U}, \text{ (for the unlabeled set } O_U\text{)}$$

where $x_i \in \mathbb{R}^D$ represents the primitives for a particular object and y^* represent the ground truth labels. We focus on the binary classification setting in which $y^* \in \{-1, 1\}$.

The primitives for each datapoint $x_i \in \mathbb{R}^D$ can be viewed as features of the data, such as individual words or patterns for text data, or numerical features such as area or perimeter of a tumor for image data (Figure 2). In our experience with collaborators using Reef, the number of primitives D is small, usually less than 10 and part of data models in weak supervision systems and open source libraries [35, 54, 32, 46]. For example, Scikit-image is equipped with functions to extract area, perimeter and other geometric properties from segmented images [46]. In our evaluation, we do not allow users to extend the set of primitives beyond those present in these data models and libraries, though they could be extended in principle.

Output Data. Reef outputs a probabilistic training label

$$\tilde{y} = P[y^* = 1] \in [0, 1]$$

for each datapoint in the unlabeled set O_U, which represents a weighted combination of the labels the different heuristics assign to each datapoint. These probabilistic training labels and associated raw data (e.g., the radiology image of a tumor from Figure 1) can then be used to train any downstream classification model, such as a convolutional neural network (CNN) [24] for image classification or a long-short term memory (LSTM) architecture [18] for natural language processing tasks. The heuristics Reef generates do not label the entire dataset and therefore cannot *generalize* to every possible datapoint they encounter. Therefore, we train an end model that can generalize beyond the primitives used to generate heuristics and assign a label to any new datapoint they see, regardless of whether the the heuristics in Reef would assign it a label.

2.2 Learning Heuristic Accuracies

Each heuristic Reef generates relies on one or more primitives and outputs a binary label or abstains for each datapoint in the unlabeled dataset (Section 3.1). If these heuristics have accuracies that are better than random chance, then taking a majority vote improves the accuracies of the predictions. However, a single bad (but prolific) voter can compromise this method. Therefore, Reef relies on existing statistical techniques, described in further detail in Section 4, that can learn the accuracies of and correlations among these noisy heuristics without using ground truth labels and assign probabilistic labels to the unlabeled dataset accordingly [36, 3, 47, 1, 38, 45]. We treat these statistical techniques as a black-box methods that learns heuristic accuracies and refer to them as *label aggregators* since they combine the labels the heuristics assign to generate a single probabilistic label per datapoint. However, Reef can generate heuristics that are much noisier than the label aggregator can handle and prevent it from operating successfully. Therefore, Reef has to determine the conditions under which the label aggregator operates succesfully, which we discuss in Section 4.

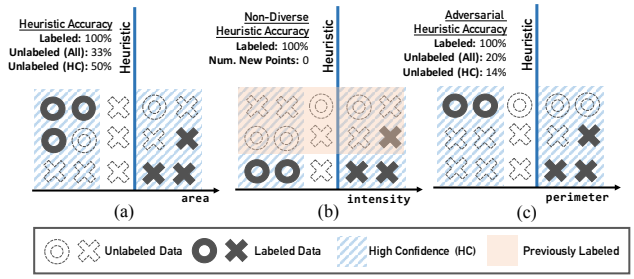

Figure 3: Heuristics with perfect accuracy on the labeled set can (a) have different accuracies on the labeled and unlabeled datasets, (b) repeatedly label the same unlabeled datapoints, and (c) be worse than random for the unlabeled set.

2.3 Reef Dataflow

The Reef process is iterative and generates a new heuristic *specialized* to the subset of the data that did not receive high confidence labels from the existing set of heuristics at each iteration. As shown in Figure 4, the three components involved in each iteration are:

1. **Synthesizer:** Based on the primitives and ground truth labels associated with the labeled dataset or its subset, the synthesizer generates a collection of heuristics, which we call the *candidate set*, that assign labels to each datapoint based on the values of its primitives (Section 3.1).

2. **Pruner:** The pruner ranks the candidate heuristics according to both its performance on the labeled set and its diversity with respect to existing heuristics for the unlabeled dataset. At each iteration, it adds the highest scoring heuristic to the existing set of heuristics, which we call the *committed set*, used to assign labels to the unlabeled data (Section 3.2).

3. **Verifier:** The verifier relies on the label aggregator to learn the accuracies of the heuristics in the committed set and assign probabilistic labels to the unlabeled data accordingly. The verifier selects the subset of the labeled data that received low confidence labels to pass to the synthesizer for the next iteration (Section 3.3).

This process is repeated until the subset the verifier passes to the synthesizer is empty, or the verifier determines that the

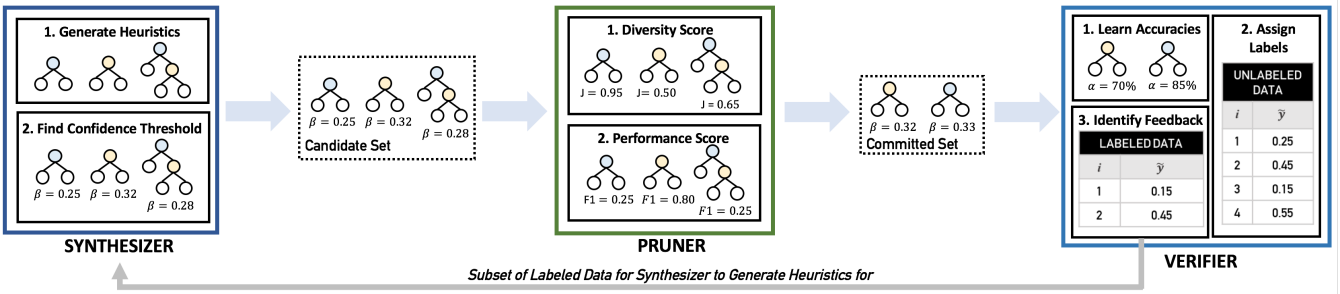

Figure 4: An overview of the Reef system. (1) The synthesizer generates a candidate set of heuristics based on the labeled dataset. (2) The pruner selects the heuristic from the candidate set to add to the committed set. (3) The verifier learns heuristic accuracies and passes appropriate feedback to the synthesizer to continue the iterative process.

conditions for the label aggregator to operate successfully are violated. We describe the statistical measure that defines this termination condition in Section 4.

2.4 Challenges

We describe the challenges associated with automating weak supervision in more detail given the pipeline described above. We discuss how each component of Reef addresses these challenges in Section 3.

Accuracy. As shown in Figure 3(a), a heuristic that has perfect accuracy for the labeled dataset can perform poorly on the unlabeled dataset. This brings up the following concerns:

(i) There is a trade-off between the accuracy and coverage of each heuristic that depends on which datapoints a heuristic assigns labels to and which it abstains for.

(ii) A heuristic can achieve perfect accuracy for the labeled dataset for multiple thresholds associated with the same primitive, but perform differently on the unlabeled dataset.

(iii) While the example in Figure 3(a) is dependent only on a single primitive value, it is possible for heuristics to rely on multiple primitives.

Diversity. As shown in Figure 3(b), a heuristic that only labels a subset of the unlabeled data can fail to assign labels to any datapoints not already labeled by heuristics in the committed set. This brings up the following concerns:

(i) The committed set can repeatedly generate heuristics that label the same subset of unlabeled data, resulting in probabilistic labels for a small portion of the unlabeled dataset.

(ii) While we can force the pruner to select heuristics that assign labels to subsets of data not previously labeled by the heuristics in the committed set, this can lead to the opposite problem of probabilistic labels from Reef covering the entire unlabeled dataset but being fairly inaccurate.

Termination Condition. As shown in Figure 3(c), a heuristic that labels a subset of the unlabeled data and has perfect accuracy on the labeled dataset can be *adversarial*, or worse than random for the same subset. This brings up the following concerns:

(i) The accuracy of heuristics on the labeled set is not informative about their accuracies on the unlabeled dataset.

(ii) While there exist label aggregators that can learn the accuracies of heuristics based only on the labels assigned to the unlabeled data, they can handle a limited amount of noise in the heuristics. However, heuristics generated automatically can easily surpass this tolerated noise limit and prevent the label aggregator from successfully learning heuristic accuracies.

3. THE REEF ARCHITECTURE

We describe the Reef system architecture, composed of the synthesizer, pruner, and verifier. We first describe how the synthesizer generates the candidate set of heuristics in Section 3.1, then discuss how the pruner selects the heuristic to add to the committed set in Section 3.2, and finally how the verifier aggregates the labels from the heuristics and passes relevant feedback to the synthesizer in Section 3.3.

3.1 Synthesizer

The Reef synthesizer takes as input the labeled set, or a subset of the labeled set after the first iteration and outputs a *candidate* set of heuristics (Figure 4). First, we describe how the heuristics are generated based on the labeled dataset and the different models the heuristic can be based on. Then, we describe how the labeling pattern of the heuristics are adjusted to assign labels to only a subset of the unlabeled dataset. Finally, we explore the trade-offs between accuracy and coverage by comparing heuristics Reef generated to other automated methods that assign labels to the entire dataset.

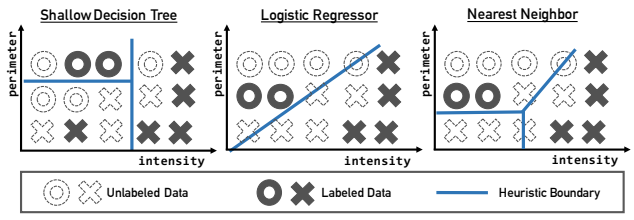

Figure 5: Heuristic models and associated boundaries.

3.1.1 Heuristic Generation

In Reef, users can select the model they want to base their heuristics on as long as the heuristic h follows the input-output form:

$$h(x_i') \rightarrow \hat{y} \in \{-1, 0, 1\}$$

where $x_i' \in \mathbb{R}^{D'}$ is a subset of primitives associated with each datapoint, $D' \leq D$ is the number of primitives in this subset, and $\hat{y}_i \in \{-1, 0, 1\}$ is either a positive or negative label or an abstention. The Reef synthesizer generates a heuristic for each possible combination of $1 \ldots D'$ primitives as shown in Algorithm 1. For example, with bag-of-words primitives, a subset of primitives represents a few specific words while a subset of bounding box attribute primitives could represent the x,y-coordinates of the bounding box. We discuss in Section 5 how $D' < 4$ for most real-world tasks.

Heuristic Models. Reef generates heuristics that are based on simple models that take as input one or more primitives and assign probabilistic labels $P[y_i^* = 1] \in [0, 1]$ to the unlabeled datapoints. We discuss how we convert these probabilistic labels to binary labels in Section 3.1.2. We focus on three specific instantiations: a decision tree (with depth limited to the number of primitives it takes as input), a logistic regressor, and a nearest neighbor model, to capture the diversity of the different decision boundaries each of them generates given the same primitive values and labeled datapoints (Figure 5). Our specific choices are motivated by the heuristics that tend to work well for different domains. For example, decision trees that rely on bag-of-words primitives represent heuristics that check whether a particular word exists. Logistic regressors and nearest neighbor models can jointly model the relationship among multiple primitives, which is especially helpful with numerical primitives common for image data.

3.1.2 Labeling Pattern Adjustment

As discussed in Section 2.4, we can improve performance of heuristics by modeling the trade-off between heuristic accuracy and coverage. Reef forces heuristics to only assign labels to datapoints they have high confidence for and abstain for the rest. To measure confidences, Reef relies on the probabilistic label $P[y_i^* = 1]$ that each heuristic model assigns to a datapoint. We define datapoints that heuristics have low confidence for as the points where $|P[y_i^* = 1] - 0.5| \leq \beta$, $\beta \in (0, 0.5)$. For each heuristic, Reef selects a threshold β that determines when a heuristic assigns a label, $\hat{y} \in \{-1, 1\}$ and when it abstains, $\hat{y} = 0$. The relation between β and $\hat{y}$ can be formally defined as:

$$\hat{y}_i = \begin{cases} 1 \text{ if } & P[\hat{y}_i = 1] \geq 0.5 + \beta \\ 0 \text{ if } & |P[\hat{y}_i = 1] - 0.5| < \beta \\ -1 \text{ if } & P[\hat{y}_i = 1] \leq 0.5 - \beta \end{cases}$$

To choose the best threshold β, we want the heuristic to assign labels to as many points as possible while being as accurate as possible for the points it labels. Therefore, we need a metric that models the trade-offs between coverage and accuracy. We calculate the precision and recall of the heuristics on the *labeled set* with N_L datapoints as a proxy to determine their performance on the unlabeled dataset, defined as

Algorithm 1: Reef Synthesis Procedure

```
function GenerateHeuristics (f, X, y*)
  Input  : Heuristic model f, Primitive matrix
           X ∈ R^{D×N_L}, Labels y* ∈ {−1, 1}^{N_L}
  Output: Candidate set of heuristics H,
           H[i] = (h(X_i'), β), Primitive comb. X_comb
for D' = 1...D do
    //generate primitive combinations of size D'
    X_comb = GenComb(X, D')
    for i = 1 ... len(X_comb) do
        X' = X_comb[i]
        h = fitFunc(f,X',y*)
        y_prob = predictProb(h,X')
        β = FindBeta(y_prob, y*)
        H[i] = (h, β)
    end
end
return H, X_comb
function FindBeta (y_prob, y*)
betaList = 0 ... 0.5
for j in len(betaList) do
    beta = betaList[j]
    F1[j] = calcF1(y*, y_prob, beta)
end
return betaList[argmax(F1)]
```

- **Precision (P)** the fraction of correctly labeled points over the total points labeled, $\frac{\sum_{i=1}^{N_L} \mathbb{1}(\hat{y}_i = y_i^*)}{\sum_{i=1}^{N_L} \mathbb{1}(\hat{y}_i \neq 0)}$
- **Recall (R)** the fraction of correctly labeled points over the total number of points, $\frac{\sum_{i=1}^{N_L} \mathbb{1}(\hat{y}_i = y_i^*)}{N_L}$
- **F1 Score** the harmonic mean of P and R, $2\frac{P \times R}{P + R}$

To balance precision and recall, the Reef synthesizer selects β for each heuristic that maximizes the F1 score on the labeled dataset O_L (Algorithm 1). The synthesizer iterates through (default 10) equally spaced values in $\beta \in (0, 0.5)$, calculates the F1 score the heuristic achieves, and selects the β value that leads to the highest F1 score. In case of ties, the synthesizer chooses the lower β value for higher coverage. In Section 5, we find that this procedure of choosing β outperforms choosing a constant β and not abstaining by up to 5.30 F1 points and 20.69 F1 points, respectively.

3.1.3 Synthesizer Tradeoffs

We explore the trade-offs that result from forcing the heuristics to abstain and how it affects end model performance for Reef compared to other automated methods like boosting, decision trees and label propagation [56], a semi-supervised learning model, all of which assign labels *to the entire unlabeled dataset*. We provide more details about these baselines in Section 5.1. We generate a synthetic experiment using one of the datasets from our evaluation, the Visual Genome dataset [23]. The associated task is to determine whether an image contains a person riding a bike given primitives associated with the bounding boxes of individual objects in the image. In this case, the heuristics use the location and size attributes of the bounding boxes as primitive inputs to the heuristics.

To study how Reef performs given varying amounts of *unlabeled* data, we set up the following simulation: given $N_L = 100$ labeled datapoints, we varied the amount of unlabeled data available to Reef from $N_U = 100$ to $N_U = 500$. Each of the methods assigned training labels to the unlabeled dataset, and this dataset was used to fine-tune the last layer of an end model, a noise-aware version of GoogLeNet [44]. We present the performance of the end model on a held-out test set in Figure 6.

Cases in which the amount of labeled and unlabeled data are the same, Reef performs worse than other methods. Since Reef only labels a portion of the unlabeled data, the end model has fewer training labels to learn from compared to the other methods that do not abstain. Since the unlabeled set is small in this situation ($N_L = N_U = 100$), the baseline methods have better end model performance since they never abstain.

Next, the performance of decision trees and boosting gets worse with increasing amounts of unlabeled data. These methods learn the same model for each datapoint plotted since they only train on $N_L = 100$ datapoints. As the amount of unlabeled data increases, these classifiers starts labeling a larger portion of the unlabeled dataset incorrectly since they never abstain. Since semi-supervised learning uses both labeled and unlabeled data to learn a classifier, it also performs better as the amount of unlabeled data increases; however, it still performs worse than Reef when the amount of unlabeled data is more than 3× larger than labeled data since semi-supervised methods also do not abstain.

As the amount of unlabeled data increases, the heuristics Reef generates continue to only assign labels with high confidence, leading to a smaller labeled training set than other methods, but high quality training labels for that portion. This is promising for machine learning applications in which the bottleneck lies in *gathering enough training labels*, while unlabeled data is readily available. Reef also outperforms baseline methods when the unlabeled data is only twice as much as labeled data (Section 5).

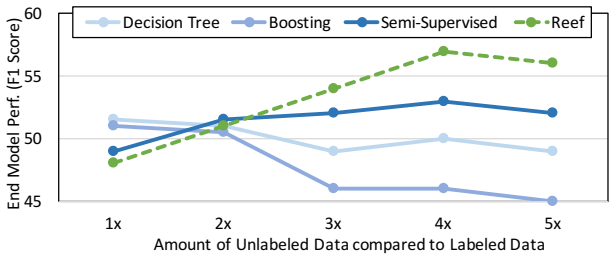

Figure 6: Linear performance increase of end model trained on labels from Reef w.r.t. unlabeled data where $1\times = 100$ datapoints.

3.2 Pruner

The pruner takes as input the candidate heuristics from the synthesizer and selects a heuristic to add to the committed set of heuristics (Figure 4). We want the heuristics in the committed set to label as much of the unlabeled data as possible, but also ensure that it performs well for the datapoints it labels in the unlabeled dataset. Since we can only measure how well heuristics perform on the small, labeled training set, we select the heuristic to add to the committed set based on both how diverse it is in terms of the datapoints it labels in the unlabeled dataset and how it performs on the small, labeled dataset.

Algorithm 2: Reef Pruning Procedure

function SelectBestHeuristic $(H, H_C, y^*, n, X_L, X_U)$
  Input : Candidate and committed set of heuristics $H, H_C, H[i] = h(X_i')$, Labels $y^* \in \{-1, 1\}^{N_L}$, Number of Assigned Labels $n \in \mathbb{Z}^+$, Primitives $X_L, X_U \in \mathbb{R}^{D \times N_L, N_U}$
  Output: Best heuristic in candidate set, $H[i] \in H$
h_{best} = None
bestScore = 0
for $h_i \in H$ **do**
  $\hat{y}_L^i = applyHeuristic(h_i, X_{i,L})$
  $f_{score} = calcF1(\hat{y}_L^i, y^*)$
  $\hat{y}_U^i = applyHeuristic(h_i, X_{i,U})$
  $j_{score} = calcJaccard(\hat{y}_U^i, n)$
  if $\frac{1}{2}(j_{score} + f_{score}) \geq bestScore$ **then**
    $h_{best} = h_i$
    bestScore = $\frac{1}{2}(j_{score} + f_{score})$
  end
end

A diverse heuristic is defined as one that labels points that have never received a label from any other heuristic. Therefore, we want to be able to maximize the *dissimilarity* between the set of datapoints a heuristic labels and the set of datapoints that previous heuristics in the committed set have already labeled. Let $n_j \in \{0, 1\}^{N_U}$ represent whether heuristic j from the candidate set has assigned labels to the datapoints in the unlabeled set. Let $n \in \{0, 1\}^{N_U}$ represent datapoints in the unlabeled set have received at least one label from the heuristics in the committed set. To measure the distance between these two vectors, we rely on the Jaccard distance metric [19], the complement of Jaccard similarity, instead of other distance metric like the l_p-norm or the Hamming distance [30], since ordering of the individual datapoints does not hold meaning. For a particular heuristic h_j in the candidate set, the generalized Jaccard distance is defined as:

$$J_j = 1 - \frac{n_j \cap n}{n_j \cup n}$$

To measure performance on the labeled dataset, Reef uses the F1 score of each heuristic in the candidate set, as defined in the previous section. As the final metric to rank heuristics, the pruner uses the average of the Jaccard distance and F1 score and selects the highest ranking heuristic from the candidate set and adds it to the committed set of heuristics. This process is described in Algorithm 2. We find that considering both performance on the labeled set and diversity on the unlabeled set improves over only considering diversity by up to 18.20 F1 points and over only considering performance by up to 6.57 F1 points in Section 5.

3.3 Verifier

With a new heuristic added to the committed set, the verifier uses the label aggregator (Section 4) to learn accuracies of the heuristics in the committed set without any ground

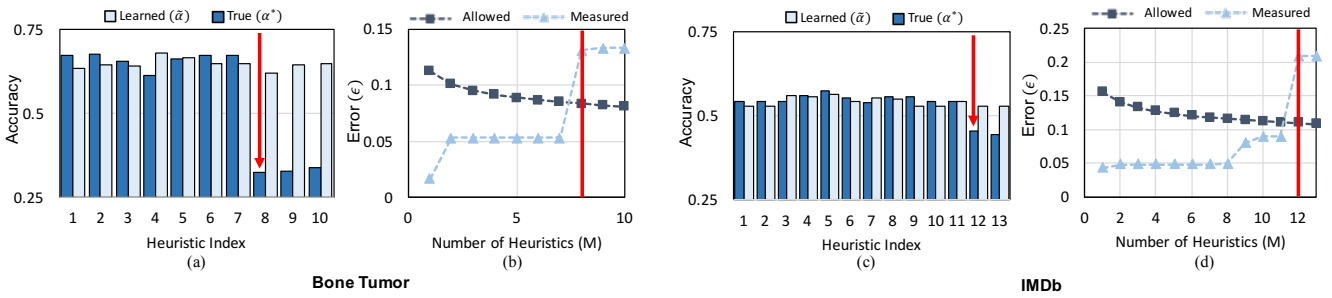

Figure 7: (a),(c) show the learned and true accuracies of the committed set of heuristics at the last iteration. (b),(d) show the allowed error and the measured error between learned and empirical accuracies across all iterations. The marked heuristic in each figure shows how Reef successfully stops generating heuristics when the new heuristic's *true accuracy* is worse than random.

truth labels and produce a single, probabilistic training label for each datapoint in the unlabeled dataset.

Algorithm 3: Reef Verifier Procedure

function FindFeedback (H_C, y^*, X_L, X_U)
  Input : Committed set of heuristics $H_C, H[i] = h(X'_i)$, Labels $y^* \in \{-1, 1\}^{N_L}$, Primitives $X_L, X_U \in \mathbb{R}^{D \times N_L, N_U}$
  Output: Subset of labeled set, O'_L
$\tilde{\alpha} =$ learnAcc (H_C, X_U)
$\hat{\alpha} =$ calcAcc (H_C, X_L, y^*)
$\tilde{y}_U =$ calcLabels $(\tilde{\alpha}, X_U)$
$\tilde{y}_L =$ calcLabels $(\tilde{\alpha}, X_L)$
if $||\tilde{\alpha} - \hat{\alpha}||_\infty \geq \epsilon$ **then**
  return $O'_L = \emptyset$
end
else
  return $o_i \in O'_L$ if $|\tilde{y}_{i,L} - 0.5| \leq \nu$
end

These probabilistic labels also represent how *confident* the label aggregator is about the labels it has assigned. For example, datapoints that have not received a single label from heuristics in the committed set will have a probabilistic label $P[y^* = 1] = 0.5$, which represents equal chance of belonging to either class. Other probabilistic labels $P[y^* = 1]$ close to 0.5 represent datapoints with low confidence, which can result from low accuracy heuristics labeling those datapoints or multiple heuristics with similar accuracies disagreeing on the labels for that datapoint. Since Reef generates a new heuristic at each iteration, we want the new heuristic to assign labels to this subset that currently has low confidence labels. However, Reef cannot generate new heuristics based on the unlabeled dataset; instead, Reef identifies datapoints *in the labeled set* that receive low confidence labels from the label aggregator. Given the label aggregator can still function given noise in the generates heuristics (Section 4), it passes this subset to the synthesizer with the assumption that similar datapoints in the unlabeled dataset would have also received low confidence labels. This process is described in Algorithm 3.

Formally, we define low confidence labels as $|\tilde{y}_i - 0.5| \leq \nu$ where $\tilde{y}$ is the probabilistic label assigned by the label aggregator and

$$\nu = \frac{1}{2} - \frac{1}{(M+1)^\eta}$$

where the $\eta > 1$ parameter (default $\eta = \frac{3}{2}$) controls the rate at which the definition of low confidence changes with increasing number of heuristics in the committed set, M. As the number of heuristics increases, we expect that fewer datapoints will have confidences near chance, or 0.5. Therefore, Reef widens the range for what probabilistic labels are considered low confidence as more heuristics are added to the committed set.

4. REEF SYSTEM GUARANTEES

We provide an overview of *generative models* [36, 47, 3, 50] that serve as the label aggregator that Reef uses to assign probabilistic labels to unlabeled datapoints in Section 4.1. As discussed in Section 2, these models can learn the accuracies and dependency structures among the noisy heuristics without using any ground truth data and can assign probabilistic labels to the unlabeled data accordingly. However, these generative models are designed to model the noise in *user-defined* heuristics, which are much less noisy than automatically generated heuristics. Specifically, the generative model assumes that heuristics always have accuracies better than 50%, an assumption that Reef-generated heuristics can violate in Section 4.2. Therefore, a key challenge in Reef is that recognizing whether the committed set includes heuristics that are worse than random for the unlabeled dataset without access to ground truth labels. To address this, we introduce a statistical measure in Section 4.3 that relies on the accuracies the generative model learns and the small labeled dataset. Finally, in Section 4.4, we formally define this statistical measure and provide a theoretical guarantee that it will recognize when the generative model is not learning heuristic accuracies successfully.

4.1 Generative Model

The simplest form of the label aggregator is a majority vote function. However, when data is labeled by a variety of sources like knowledge bases in distant supervision and user-defined heuristics, the more accurate approach is to learn and model the different accuracies of these labeling sources to combine their labels [10, 36]. For the Reef setting, we could also rely on the accuracies of the heuristics on the labeled dataset, $\hat{\alpha}$; however, this would fail to

take advantage of the information present in the unlabeled dataset and degrade end model performance by up to 8.43 F1 points (Section 5). Therefore, Reef relies on our previous work on generative models that can learn accuracies for the heuristics using their labeling patterns on the unlabeled dataset, without using any ground truth labels [47, 36, 50].

Formally, the goal of the generative model is to estimate the true accuracies of the heuristics, $\alpha^* \in \mathbb{R}^M$, using the labels the heuristics assign to the unlabeled data, $\hat{Y} \in \{-1, 0, 1\}^{M \times N_U}$. It models the true class label $Y^* \in \{-1, 1\}^{N_U}$ for a datapoint as a latent variable in a probabilistic model and in the simplest case, assumes that each labeling source is independent. The generative model is expressed as a factor graph:

$$\pi_\phi(\hat{Y}, Y^*) = \frac{1}{Z_\phi} \exp\left(\phi^T \hat{Y} Y^*\right) \quad (1)$$

where Z_ϕ is a partition function to ensure π is a normalized distribution. The parameter $\phi \in \mathbb{R}^M$ is used to calculate the learned accuracies $\tilde{\alpha} = \frac{\exp(\phi)}{1+\exp(\phi)} \in \mathbb{R}^M$ (defined pointwise). It is estimated by maximizing the marginal likelihood of the observed heuristics $\hat{Y}$, using a method similar to contrastive divergence [17], alternating between using stochastic gradient descent and Gibbs sampling [36, 3]. The generative model assigns *probabilistic training labels* by computing $\tilde{Y} = \pi_\phi(Y^* \mid \hat{Y})$ for each datapoint.

These probabilistic training labels can then be used to train any machine learning end model of choice with *noise-aware* loss [35, 36], as defined by

$$\min \sum_{i=1}^{N_U} \mathbb{E}_{y \sim \tilde{Y}} \left[l(h_\theta(o_i), y) \right]$$

where $o_i \in O_U$ is an object in the unlabeled dataset and $\tilde{Y}$ are the probabilistic training labels. In our evaluation, we adjust the loss functions of several popular machine learning models to the use the noise-aware variant when training on probabilistic labels.

4.2 Assumption Violation

Since the generative model requires no ground truth labels to learn heuristic accuracies $\tilde{\alpha}$, it has to solve an underdetermined problem where the heuristics could have accuracies $\tilde{\alpha}$ or $1 - \tilde{\alpha}$. Therefore, the generative model assumes that the labeling sources always perform better than random, that is $\alpha^* > 0.5$. Since the model was initially designed to estimate accuracies for user-defined sources of labels like heuristics or knowledge bases, this was a reasonable assumption that was rarely violated [36, 3, 47].

Since Reef generates these heuristics automatically, it is possible for the heuristics be accurate for the labeled set but adversarial for the unlabeled data. This violates the generative model's assumption that $\alpha^* > 0.5$ and can therefore deteriorate the quality of training labels the generative model assigns to the unlabeled data. An example of such a situation is shown in Figure 7(a),(c) for two real datasets. The 8th and 12th heuristics, respectively, have an accuracy worse than 50% on the unlabeled dataset. However, since the generative model does not know that this assumption has been violated, it *learns* an accuracy much greater than 50% in both cases. If these heuristics are included in the generative model, the generated probabilistic training labels affect end model performance by 5.15 F1 and 4.05 F1 points compared to not including these heuristics. Therefore, there is a need for determining when the generated heuristics are too noisy for the generative model to learn accuracies for, without access to the true accuracies of the heuristics or ground truth labels for the unlabeled dataset.

4.3 Statistical Measure

Reef can take advantage of the *small, labeled dataset* to indirectly determine whether the generated heuristics are worse than random for the unlabeled dataset or not. We define the empirical accuracies of the heuristics as

$$\hat{\alpha}_i = \frac{1}{N_i} \sum_{\hat{Y}_{ij} \in \{-1,1\}} \mathbb{1}(\hat{Y}_{ij} = Y_j^*),$$

for $i = 1 \ldots M$. $\hat{Y}_{ij} \in \{-1, 0, 1\}$ is the label heuristic i assigned to the j-th datapoint in the labeled set O_L, and N_i is the number of datapoints where $\hat{Y}_i \in \{1, -1\}$.

Our goal is to use the empirical accuracies, $\hat{\alpha}$ to estimate whether the learned accuracies, $\tilde{\alpha}$ are close to the true accuracies, α^*. We define the generative model learning accuracies successfully as $||\alpha^* - \tilde{\alpha}||_\infty < \gamma$. This translates to the maximum absolute difference between the learned and true accuracies being less than γ, a positive constant to be set. Then, we define the measured error between the learned and empirical accuracies as $||\hat{\alpha} - \tilde{\alpha}||_\infty$. To guarantee with high probability that the generative model learns accuracies within some constant γ, we want to find ϵ, the largest allowed error between the learned and empirical accuracies, $||\hat{\alpha} - \tilde{\alpha}||_\infty \leq \epsilon$. We discuss the exact form of ϵ in Section 4.4.

In Reef, we compare the measured error $||\hat{\alpha} - \tilde{\alpha}||_\infty$ to the calculated value of ϵ at *each iteration*, as shown in Figure 7(b),(d). If the measured error is greater than ϵ, then we stop the iterative process of generating heuristics and use the probabilistic training labels generated at the previous iteration (since the heuristic generated at the current iteration led to measured error being greater than ϵ). As shown in Figure 7, this stopping point maps to the iteration at which the new heuristic generated has a true accuracy, α^*, worse than 50% for the unlabeled dataset (we only calculate α^* for demonstration since we would not have access to ground truth labels for the dataset for real use cases). This procedure assumes that once the synthesizer generates a heuristic that is worse than random for the unlabeled dataset, it will never generate heuristics that will be helpful in labeling the data anymore. Empirically, we observe that this is indeed the case as shown for two real tasks in Figure 7(a) and (c).

4.4 Theoretical Guarantees

Assuming that the objects in the labeled set O_L are independent and identically distributed, we provide the following guarantee on the probability of the generative model learning the accuracies successfully:

Proposition 1: *Suppose we have M heuristics with empirical accuracies $\hat{\alpha}$, accuracies learned by the generative model $\tilde{\alpha}$, and measured error $||\hat{\alpha} - \tilde{\alpha}||_\infty \leq \epsilon$ for all M iterations. Then, if each heuristic labels a minimum of*

$$N \geq \frac{1}{2(\gamma - \epsilon)^2} \log\left(\frac{2M^2}{\delta}\right)$$

datapoints at each iteration, the generative model will succeed in learning accuracies within $||\alpha^* - \tilde{\alpha}||_\infty < \gamma$ across all

Application	Domain	N_L	N_U	$\frac{N_U}{N_L}$	D	Label Source	Task
Bone Tumor	Image	200	400	2.0	17	DT + User	Aggressive vs. Non-aggressive Tumor
Mammogram	Image	186	1488	8.0	10	UDF	Malignant vs. Benign Tumor
Visual Genome	Image	429	903	2.1	7	UDF	Identify 'person riding bike'
MS-COCO	Multi-Modal	6693	26772	4.0	105	UDF	Identify whether person in image
IMDB	Text	284	1136	4.0	322	UDF/Crowd	Action vs. Romance Plot Descriptions
Twitter	Text	123	14551	118.3	201	Crowd	Positive vs. Negative Tweets
CDR	Text	888	8268	9.3	298	UDF + DS	Valid vs. Invalid chemical-disease relation

Table 1: Dataset Statistics and Descriptions. N_L, N_U are size of labeled and unlabeled datasets. $\frac{N_L}{N_U}$: ratio of unlabeled to labeled data, D: number of primitives. Label sources are previous sources of training labels (DT: decision tree, UDF: user-defined functions, DS: distant supervision with knowledge base.)

iterations with probability $1 - \delta$.
We provide a formal proof for this proposition in the [49].

We step through the above proposition here: we require each heuristic to assign labels to at least N datapoints to guarantee that the generative model will learn accuracies within γ of the true accuracies, given the measured error is less than ϵ *for all iterations.* The bound in Proposition 1 relies on M, the number of heuristics. To adjust Proposition 1 to solve for the maximum allowed error ϵ *at each iteration*, we remove a factor of M from the expression:

$$\epsilon = \gamma - \sqrt{\frac{1}{2N} \log\left(\frac{2M}{\delta}\right)}.$$

This value is plotted in Figure 7(b) and (d) against the value of the measured error $||\hat{\alpha} - \tilde{\alpha}||_\infty$. Reef stops generating new heuristics as soon as the measured error surpasses the allowed error.

The above proposition relies only on the measured error to guarantee whether the generative model is learning accuracies successfully, which means that we do not have to have *any* assumption about the heuristics, such as about their true accuracies on the labeled set or how they are correlated with each other. Next, we only assume that the small labeled set is i.i.d. and have no assumptions on the large, unlabeled training set. In Reef, users can easily fulfill the requirement of an i.i.d labeled set since it is as much as 100x smaller than the unlabeled set and not worry about the distribution of the unlabeled data, translating to significantly lower effort for manually shaping the distribution of the larger, unlabeled set.

5. EVALUATION

To evaluate Reef, we compare the performance of end models like CNNs or LSTMs trained on labels generated by Reef and other baseline methods. We seek to experimentally validate the following claims:

- **Training labels from Reef outperform labels from automated baseline methods** We compare Reef to models that generate heuristics using only the labeled data, such as boosting and decision trees, and semi-supervised methods, which utilize both labeled and unlabeled datasets. Reef outperforms the above methods by up to 14.35 F1 points. We also compare to transfer learning using only the labeled dataset for select tasks, which Reef outperforms by up to 5.74 F1 points.
- **Training labels from Reef outperform those from user-developed heuristics** We compare the performance of heuristics generated by Reef to heuristics developed by users. We show that Reef can use the *same* amount of labeled data as users to generate heuristics and assign training labels to unlabeled data and improve end model performance by up to 9.74 F1 points.
- **Each component of Reef boosts overall system performance** We evaluate the separate components of the Reef system by comparing end model performance by changing how the β parameter is chosen in the synthesizer, how the pruner selects a heuristic to add to the committed set, and different label aggregation methods in the verifier. Compared to the complete Reef system, we observe that performance can degrade by up to 20.69 F1 points by removing these components.

5.1 Experiment Setup

We describe the datasets and baseline methods used to evaluate Reef. We then discuss implementation details and performance metrics used for these tasks.

5.1.1 Datasets

To evaluate Reef, we consider real-world deployments and tasks on open source datasets in domains like image and text classification, sentiment analysis, and multi-modal classification. For each of the tasks, previous techniques to assigns training labels included using crowdsourcing, user-defined functions, and decision trees based on a small, labeled dataset. Summary statistics are provided in Table 1 and additional details are in [49].

Image Classification. We first focus on two real-world medical image classification tasks that we collaborated on with radiologists at Stanford Hospital and Clinics. The **Bone Tumor** and **Mammogram** tumor classification to demonstrate how Reef generated heuristics compared to those developed by domain experts. We use the bone tumor dataset to show how Reef performs with a set of domain-specific primitives that are difficult for radiologists to interpret. We use the mammogram tumor classification set to evaluate Reef when the quality of the primitives is mediocre and not domain-specific. Working with graduate students in the Stanford Computer Vision lab, we defined a task to identify

Application	Reef F1 Score	Improvement over Automated Baselines				Improvement over User-Driven Methods
		DT	Boosting	Transfer Learning	Semi-Supervised	
Bone Tumor	71.55	9.35	8.65	-	6.77	9.13
Mammogram	74.54	5.61	5.02	5.74	3.26	9.74
Visual Genome	56.83	8.83	6.2	5.58	5.94	6.58
MS-COCO	69.52	2.77	2.7	2.51	1.84	2.79
IMDb	62.47	9.56	12.12	3.36	14.35	3.67
Twitter	78.84	5.92	4.43	-	3.84	13.8
CDR	41.56	9.68	11.22	-	7.49	-7.71

Table 2: Absolute performance of Reef and improvement over automated baselines and user-defined heuristics in terms of F1 Score of the end model. DT: Decision Tree

images of "person riding bike" to see how Reef can help automatically generate training sets for complex relations in images. We relied on the **Visual Genome** database [23] with bounding box characteristics as primitives and also study how Reef performs with severe class imbalance.

Text and Multi-Modal Classification. Moving from the image domain where the number of primitives was limited due to the lack of interpretability of the building blocks of images, we applied Reef to text and multi-modal datasets to study how well Reef operated in domains where humans could easily interpret and write rules over the raw data. For all text-based tasks, we generated primitives by featurizing the text using a bag-of-words representation. The **MS-COCO** dataset [27] had heuristics generated over captions and classification performed over associated images, and the **IMDb** plot summary classification was purely text-based. The **Twitter** sentiment analysis dataset relied on crowdworkers for labels [28] while the chemical-disease relation extraction task (**CDR**) [53] relied on external sources of information like knowledge bases and user-defined heuristics.

5.1.2 Baseline Methods

We compare Reef against other automated methods that can use a small labeled and large unlabeled dataset to generate training labels for the unlabeled dataset. We also compare against user-driven labeling methods to generate training labels.

Decision Tree [39]: We pass in all the primitives associated with the labeled datapoints and do not limit the maximum depth for the tree, allowing the tree to be as complex as necessary. This baseline does not use the unlabeled dataset and represents generating a single heuristic using only the labeled dataset.

Boosting [15]: We compare to AdaBoost, which generates a collection of weak classifiers, which are imperfect decision trees in our evaluation. This method also takes as input all the primitives associated with the labeled datapoints for generating the imperfect decision trees. This baseline does not use the unlabeled dataset and represents generating a multiple, noisy heuristics using only the labeled dataset.

Semi-Supervised Learning [56]: We compare to label spreading, a semi-supervised method that uses *both* the labeled and unlabeled dataset to assign training labels. This baseline represents generating a single 'heuristic' in the form of a black-box model using both the labeled and unlabeled datasets.

Transfer Learning: For select tasks, there exist pre-trained models that performed well on a similar task for the same data modality. In these cases, we also compared against transfer learning, or fine-tuning the weights from a pre-trained model using only the labeled datapoints. For IMDb and Twitter, we use GLoVE embeddings [33] and only learn weights for a single LSTM layer on top of the embeddings, only tune the last layer of a VGGNet [41] for MS-COCO, and tune the weights of a GoogLeNet [44] pre-trained on ImageNet [11] over a few iterations for the Visual Genome and Mammogram datasets. This baseline represents a popular method that users rely on when they have a small amount of labeled data.

User-Driven Methods: As shown in Table 1, training labels for all seven tasks were previously generated by some user-driven labeling method, such as user-defined heuristics, relying on distant supervision via external knowledge bases, or crowdsourcing. These were developed by users, ranging from domain experts to machine learning practitioners and input to label agregators we developed [47, 35, 48]. For tasks like CDR, bone tumor, and mammogram that required specific domain knowledge, the time taken for bioinformatics experts and radiologists to manually develop heuristics ranged from a few days to a few weeks. For tasks that did not require domain expertise, such as IMDb and Visual Genome, graduate students wrote a small number of heuristics over a period of a few hours. In both cases, users encoded their domain knowledge in heuristics, evaluated their performance on a small, held-out labeled set, and iteratively improved upon the heuristics.

5.1.3 Implementation Details

Primitives for Reef Since text-based tasks used a bag-of-words representation, the primitives were fairly sparse. To ensure that each generated heuristic could assign a label to at least one datapoint in the unlabeled set, we filtered the primitives so that they represented words that were present in *both* the labeled and unlabeled dataset. We further filtered the primitives and only used the primitives that are active for at least 5% of the unlabeled and labeled datasets. For image-based tasks, we found that Reef never generated heuristics that relied on more than 4 primitives as input, while for text-based tasks, it only generated heuristics that relied on a single primitive. We hypothesize that the simplicity of generated heuristics is a result of complex heuristics easily overfitting to the labeled dataset, not assigning enough labels to unlabeled datapoints, and therefore not being selected by the pruner. The relatively small number of

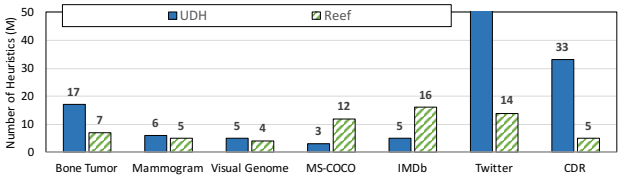

Figure 8: For image domains, Reef generates fewer heuristics than users and the opposite is true for text domains. Twitter uses 320 crowdworkers.

primitives heuristics used as input led to a maximum runtime of 5.43 mins, which was for the MS-COCO task.

Performance Metrics To measure performance, we report the F1 score on a test set of an end model trained on labels from Reef and the baseline methods described earlier in this section. We report F1 score instead of accuracy since some datasets are have class imbalance and that can lead to high accuracy despite poor performance. The F1 scores for the end model are defined in terms of true positives, which is different from the definition provided in Section 3.1.2, since the end models never abstain. For image classification tasks, we use popular deep learning models like GoogLeNet and VGGNet that take the raw image as input, while for text tasks we use a model composed of a single embedding and a single LSTM layer that take the raw text sentence(s) as input. These models take as input the probabilistic or binary training labels from Reef or the baseline methods and minimize the noise-aware loss, as defined in Section 4.

5.2 End to End System Comparison

We validate that Reef generates training labels that outperform those from automated baseline methods and those from user-driven labeling methods on the seven tasks in Section 5.1.

5.2.1 Automated Methods

Table 2 shows that Reef can outperform automated baseline methods by up to 14.35 F1 points on the seven tasks from Section 5.1. As expected, Reef outperforms decision trees, which fit a single model to the labeled dataset, by 7.38 F1 points on average, the largest improvement compared to other baselines. The method that performs the closest to Reef for most tasks is semi-supervised learning, which takes advantage of both the unlabeled and unlabeled dataset, but fails to account for diversity, performing worse than Reef by 6.21 F1 points on average. Finally, compared to transfer learning which does not have to learn a representation of the data from scratch, Reef performs up to 5.74 F1 points better using the same amount of labeled data. This demonstrates how for many tasks, using a larger training set with noisy labels is able to train a better end model from scratch than fine tuning a pre-trained model with a small labeled dataset.

5.2.2 User-Driven Labeling Methods

For each of the tasks from Section 5.1, we compare end model performance trained on labels Reef generates to labels from heuristics or other manually generated labeling sources in Table 2. In Figure 8, we compare the number of heuristics Reef generated versus how many user developed. We discuss results for the image and text domain below.

Image Domain. For image domains (Bone Tumor, Mammogram, Visual Genome), Reef generates fewer heuristics than users since each heuristic relies on two or more primitives, while user-defined heuristics usually relied on one or two primitives (Figure 8). Therefore, Reef heuristics are able to exploit more information using a single heuristic. Next, the primitives for image domains are numerical, such as area and perimeter, and associated heuristics had to find the correct numerical threshold. For user-defined heuristics, this process of choosing the optimal threshold was based on iteratively checking performance on the labeled dataset for *each* primitive used. On the other hand, Reef is able to set this optimal threshold automatically since it trains heuristics on the small, labeled dataset and adjusts the thresholds based on the β parameter. Therefore, Reef outperforms user-defined heuristics by up to 9.74 F1 points in the image domain while generating fewer heuristics.

Text Domain. For tasks where the underlying primitives were based on a bag-of-words representation (MS-COCO, IMDb, Twitter, CDR), Reef relies on a large number of heuristics, almost 5x the number that users developed for MS-COCO and IMDb (Figure 8). While users could rely on complex text-based patterns to develop their heuristics, Reef-generated heuristics rely on a *single* primitive. Heuristics that used a larger number of primitives, or large number of words due to the bag-of-words representation, had fairly low coverage and were never picked by the pruner due to their lack of diversity. Therefore, Reef generates several, simple heuristics to achieve similar performance to complex, user-defined heuristics.

Despite this apparent setback, Reef outperforms user-defined heuristics by up to 13.80 F1 points for text tasks, as shown in Table 2. Note that it performed 7.71 F1 points worse on the CDR task. The user-defined heuristics relied on distant supervision via the Comparative Toxicogenomics Database [9], which Reef did not have access to. This demonstrates a drawback of Reef since it can only generate heuristics based on the information present in the small labeled dataset and associated primitives and not incorporate external information.

5.3 Micro-Benchmarking Results

We evaluate the individual components of the Reef system and show how changing the design choices behind each component can affect end model performance by up to 20.69 F1 points.

5.3.1 Synthesizer

First, we compare how the different heuristic types perform for select tasks in Table 3 and show how much better the best heuristic type (marked as 0) performs compared to alternate heuristic types. For both text-based tasks, decision tree and perceptron based heuristics perform the same since they both rely on a single primitive and learn the same threshold to make a binary decision. These heuristic forms essentially check whether a word exists in a sentence or not. The nearest neighbor approach could break ties among the K neighbors arbitrarily, which leads to worse performance on text datasets than the other two heuristics forms.

Next, we set $\beta = 0$ to prevent heuristics from abstaining and set it to a constant $\beta = 0.25$, the midpoint of possible

	Improvement Over				
Dataset	**DT**	**LR**	**NN**	$\beta = 0$	$\beta = 0.25$
Bone Tumor	2.73	0.00	4.62	2.35	3.77
Visual Genome	3.22	3.38	0.00	7.99	5.30
MS-COCO	0.00	0.00	0.24	2.51	2.51
IMDb	0.00	0.00	14.32	20.69	2.13

Table 3: Improvement of best heuristic type over others and Reef choosing β over never abstaining ($\beta = 0$) and midpoint value ($\beta = 0.25$). 0.00 is best heuristic type that was best for each task. DT: decision tree, LR: logistic regressor; NN: nearest neighbor.

values $\beta \in (0, 0.5)$ (Table 3). We demonstrate how forcing heuristics to abstain can improve performance by up to 20.69 F1 points. Next, we show that *choosing* the correct β value instead of arbitrarily abstaining for some low confidence datapoints can improve end model performance by up to 5.30 F1 points.

5.3.2 Pruner

We show the performance of the pruner compared to only optimizing for either performance (with F1 score) or diversity (with Jaccard distance) in Table 4. For text tasks, only optimizing for performance comes within 2.15 F1 points of the Reef pruner since each heuristic selecting a different word automatically accounts for diversity. On the other hand, only optimizing for diversity in text domains can affect performance by up to 18.20 F1 points since it could result in a large portion of the unlabeled dataset receiving low-quality labels.

	Improvement Over	
Dataset	**F1 Only**	**Jaccard Only**
Bone Tumor	3.86	8.84
Visual Genome	6.57	7.33
MS-COCO	2.51	18.20
IMDb	2.15	14.23

Table 4: Reef pruner optimizing for performance (F1) and diversity (Jaccard) jointly compared to optimizing for either.

5.3.3 Verifier

Finally, we look at how learning accuracies for the heuristic to aggregate their labels compares to majority vote in Table 5. In the text domains where the number of heuristics generated is more than 15, the majority vote score comes within 1.80 F1 points of the Reef verifier. With a large number of heuristics, each datapoint receives enough labels that learning accuracies helps change the final label only a few times [25].

We compare performance to using the empirical accuracies of the heuristics, $\hat{\alpha}$, rather than learn accuracies based on labels assigned to the unlabeled data. This method performs worse than the Reef verifier by up to 8.43 F1 points, which demonstrates the necessity of the label aggregator *learning* the accuracies of the heuristics from the labels they assign to the unlabeled data.

	Improvement Over		
Dataset	**Empirical**	**No Term.**	**MV**
Bone Tumor	5.42	5.15	3.78
Visual Genome	8.43	7.09	6.59
MS-COCO	7.98	3.70	3.00
IMDb	5.67	4.05	1.80

Table 5: Reef verifier aggregation compared to Empirical: using $\hat{\alpha}$ instead of $\tilde{\alpha}$, No Term: no termination condition, MV: majority vote across labels

Finally, we look at how end model performance would be affected if we generated heuristics iteratively without the termination condition. We generate heuristics till there are no more datapoints in the small, labeled dataset with low confidence labels and find that this can degrade end model performance by up to 7.09 F1 points as shown in Table 5.

6. RELATED WORK

We provide an overview of automated methods for labeling data based on heuristics and methods that use both labeled an unlabeled data. We also discuss other methods to aggregate noisy sources of labels.

Heuristic-Based Labeling. The inspiration for Reef comes from program synthesis, where programs are generated given access to a set of input-output pairs [14, 42], reference implementations [2], or demonstrations [20]. Specifically, the Reef design is based loosely on counter-example guided inductive synthesis (CEGIS) in which a synthesizer generates programs, passes it to the verifier that decides whether the candidate program satisfies the given specifications, and passes relevant feedback to the synthesizer [42, 14, 20, 43]. However, unlike Reef, such models only synthesize programs that match *all* the specified input-output pairs. Other works also generate heuristics to help interpret the underlying data labels [52, 51], but neither methods use *unlabeled data* since the programs generated either mimic the desired program perfectly or provide interpretations for existing labels.

Focusing on the problem of generating training data, Snorkel [35] is a system that relies on domain experts manually developing heuristics, patterns, or distant supervision rules to label data noisily. While users in Snorkel rely on a small, labeled dataset to evaluate and refine their heuristics, Reef automatically generates heuristics using the labeled and unlabeled data it has access to. Snorkel uses the generative model to aggregate heuristic labels, but Reef can generate heuristics that are noisier than the generative model can account for. Therefore, it uses a statistical measure to determine when the generative model can be used (Section 4). Other methods that rely on imperfect sources of labels that are partially user-defined include distant supervision [7, 29], which relies on information present in knowledge bases and heuristic patterns [16, 5] that focus on specific tasks like relation and information extraction.

Utilizing Labeled and Unlabeled Data. To train a deep learning model with a small, labeled dataset, a common approach is using transfer learning, or retraining models that have been trained for different tasks that have abundant training data in the same domain [31]. However, this approach does not take advantage of any unlabeled data avail-

able. Semi-supervised learning leverages both labeled and unlabeled data, along with assumptions about low-dimensional structure and smoothness of the data to automatically assign labels to the unlabeled data [6, 56]. Unlike semi-supervised learning, which generates a single black-box model, Reef generates multiple, diverse heuristics to label the unlabeled data. Moreover, as demonstrated in Section 5, Reef performs better than a specific semi-supervised model, label spreading [56], when the amount of unlabeled data is larger than than the amount of labeled data. Co-training [4] also take advantage of both labeled and unlabeled data and trains two *independent* models on two separate views of the data. Reef does not require access to separate feature sets as views and can generate more than two heuristics (classifiers) that can be correlated with each other (Section 4).

Combining Noisy Labels. Combining labels from multiple sources like heuristics is well-studied problem [10], especially in the context of crowdsourcing [8, 21, 55]. However, these methods assume the labeling sources are not generated automatically and requires a labeled dataset to *learn* the accuracies of the different sources. Other methods, including our previous work [47, 36, 50], rely on generative models to learn accuracies and dependencies among labeling sources [1, 38, 45]. However, these models assume that the labeling sources are never worse than random. Reef automatically determines when this assumption does not hold by using the small, labeled dataset (Section 4).

Areas like data fusion [12, 37, 34] and truth discovery [26] also look at the problem of estimating how reliable different data sources are while utilizing probabilistic graphical models like Reef. However, the labeling sources are generated automatically in Reef to label user-provided data and can either label or abstain for each datapoint.

7. CONCLUSION

Reef is the first automated weak supervision system that takes as input a small labeled dataset and a large unlabeled dataset to assign training labels to the unlabeled data, which can be used to train a downstream model of choice. It iteratively generates heuristics that are accurate and diverse for the unlabeled dataset using the small, labeled dataset. Reef relies on a statistical measure to determine when generated heuristics are too noisy and therefore when to terminate the iterative process. We demonstrate how training labels from Reef outperform labels from semi-supervised learning by up to 14.35 F1 points and from user-defined heuristics by up to 9.74 F1 points in terms of end model performance for tasks across various domains. Our work suggests that there is potential to utilizing a small amount of labeled data to make the process of generating training labels much more efficient.

8. REFERENCES

[1] E. Alfonseca, K. Filippova, J.-Y. Delort, and G. Garrido. Pattern learning for relation extraction with a hierarchical topic model. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 54–59. Association for Computational Linguistics, 2012.

[2] R. Alur, R. Bodik, G. Juniwal, M. M. Martin, M. Raghothaman, S. A. Seshia, R. Singh, A. Solar-Lezama, E. Torlak, and A. Udupa. Syntax-guided synthesis. In *Formal Methods in Computer-Aided Design (FMCAD), 2013*, pages 1–8. IEEE, 2013.

[3] S. H. Bach, B. He, A. Ratner, and C. Ré. Learning the structure of generative models without labeled data. *arXiv preprint arXiv:1703.00854*, 2017.

[4] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[5] R. Bunescu and R. Mooney. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 576–583, 2007.

[6] O. Chapelle, B. Scholkopf, and A. Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[7] M. Craven, J. Kumlien, et al. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999, pages 77–86, 1999.

[8] N. Dalvi, A. Dasgupta, R. Kumar, and V. Rastogi. Aggregating crowdsourced binary ratings. In *Proceedings of the 22nd international conference on World Wide Web*, pages 285–294. ACM, 2013.

[9] A. P. Davis, C. J. Grondin, R. J. Johnson, D. Sciaky, B. L. King, R. McMorran, J. Wiegers, T. C. Wiegers, and C. J. Mattingly. The comparative toxicogenomics database: update 2017. *Nucleic acids research*, 45(D1):D972–D978, 2016.

[10] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[12] X. L. Dong and D. Srivastava. Big data integration. *Synthesis Lectures on Data Management*, 7(1):1–198, 2015.

[13] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[14] S. Gulwani. Synthesis from examples: Interaction models and algorithms. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2012 14th International Symposium on*, pages 8–14. IEEE, 2012.

[15] T. Hastie, S. Rosset, J. Zhu, and H. Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[16] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992.

[17] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural*

computation, 14(8):1771–1800, 2002.

[18] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] P. Jaccard. Lois de distribution florale dans la zone alpine. *Bull Soc Vaudoise Sci Nat*, 38:69–130, 1902.

[20] S. Jha, S. Gulwani, S. A. Seshia, and A. Tiwari. Oracle-guided component-based program synthesis. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 215–224. ACM, 2010.

[21] M. Joglekar, H. Garcia-Molina, and A. Parameswaran. Comprehensive and reliable crowd assessment algorithms. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 195–206. IEEE, 2015.

[22] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1988–1997. IEEE, 2017.

[23] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[25] H. Li, B. Yu, and D. Zhou. Error rate analysis of labeling by crowdsourcing. In *ICML Workshop: Machine Learning Meets Crowdsourcing. Atalanta, Georgia, USA*. Citeseer, 2013.

[26] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han. A survey on truth discovery. *ACM Sigkdd Explorations Newsletter*, 17(2):1–16, 2016.

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[28] C. Metz. Crowdflower dataset: Airline twitter sentiment, 2015. `https://www.crowdflower.com/data/airline-twitter-sentiment/`.

[29] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

[30] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov. Hamming distance metric learning. In *Advances in neural information processing systems*, pages 1061–1069, 2012.

[31] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[33] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[34] R. Pochampally, A. Das Sarma, X. L. Dong, A. Meliou, and D. Srivastava. Fusing data with correlations. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 433–444. ACM, 2014.

[35] A. J. Ratner, S. H. Bach, H. R. Ehrenberg, and C. Ré. Snorkel: Fast training set generation for information extraction. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1683–1686. ACM, 2017.

[36] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré. Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems*, pages 3567–3575, 2016.

[37] T. Rekatsinas, M. Joglekar, H. Garcia-Molina, A. Parameswaran, and C. Ré. Slimfast: Guaranteed results for data fusion and source reliability. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1399–1414. ACM, 2017.

[38] B. Roth and D. Klakow. Combining generative and discriminative model scores for distant supervision. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 24–29, 2013.

[39] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[40] R. Sawyer-Lee, F. Gimenez, A. Hoogi, and D. Rubin. Curated breast imaging subset of ddsm, 2016.

[41] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[42] R. Singh and S. Gulwani. Synthesizing number transformations from input-output examples. In *International Conference on Computer Aided Verification*, pages 634–651. Springer, 2012.

[43] A. Solar-Lezama, L. Tancau, R. Bodik, S. Seshia, and V. Saraswat. Combinatorial sketching for finite programs. *ACM Sigplan Notices*, 41(11):404–415, 2006.

[44] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[45] S. Takamatsu, I. Sato, and H. Nakagawa. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 721–729. Association for Computational Linguistics, 2012.

[46] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[47] P. Varma, B. D. He, P. Bajaj, N. Khandwala, I. Banerjee, D. Rubin, and C. Ré. Inferring generative model structure with static analysis. In *Advances in Neural Information Processing Systems*, pages 239–249, 2017.

[48] P. Varma, D. Iter, C. De Sa, and C. Ré. Flipper: A systematic approach to debugging training sets. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, page 5. ACM, 2017.

[49] P. Varma and C. Ré. Reef: Automating weak supervision to label training data, 2018. `https://paroma.github.io/tech_report_reef.pdf`.

[50] P. Varma, R. Yu, D. Iter, C. De Sa, and C. Ré. Socratic learning: Correcting misspecified generative models using discriminative models. *arXiv preprint arXiv:1610.08123*, 2017.

[51] F. Wang and C. Rudin. Falling rule lists. In *Artificial Intelligence and Statistics*, pages 1013–1022, 2015.

[52] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille. Or's of and's for interpretable classification, with application to context-aware recommender systems. *arXiv preprint arXiv:1504.07614*, 2015.

[53] C.-H. Wei, Y. Peng, R. Leaman, A. P. Davis, C. J. Mattingly, J. Li, T. C. Wiegers, and Z. Lu. Overview of the biocreative v chemical disease relation (cdr) task. In *Proceedings of the fifth BioCreative challenge evaluation workshop*, pages 154–166. Sevilla Spain, 2015.

[54] S. Wu, L. Hsiao, X. Cheng, B. Hancock, T. Rekatsinas, P. Levis, and C. Ré. Fonduer: Knowledge base construction from richly formatted data. *arXiv preprint arXiv:1703.05028*, 2017.

[55] Y. Zhang, X. Chen, D. Zhou, and M. I. Jordan. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. In *Advances in neural information processing systems*, pages 1260–1268, 2014.

[56] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328, 2004.

APPENDIX

A. EVALUATION DATASETS

We describe each task described in Section 5 below.

Bone Tumor The first dataset in the medical domain consists of 800 X-rays of bones from various parts of the body, and we give Reef access to 200 hand-labeled examples. Each X-ray also contains an associated binary mask which denotes where the tumor occurs in the image. Radiologists extract 400 domain-specific primitives based on the shape, texture, and intensity of the tumor, out of which we use 17 as primitives for Reef and the rest as features for a logistic regression model. *We use this dataset to show how Reef can efficiently assign labels to previously unlabeled images and improve over the oracle score.*

Mammogram The second medical dataset is based on the publicly available DDSM-CBIS [40] dataset, in which each CT scan has an associated segmentation binary mask that outlines the location of the tumor in the mammogram and a label for whether the tumor is malignant or benign. Given this segmentation, we extract simple features such as area and perimeter, which require no domain expertise, unlike the Bone Tumor dataset. *We use this dataset to demonstrate how Reef performs well even when the quality of the primitives is mediocre and not domain-specific.*

Visual Genome [23] We define a image query of whether an image contains a person riding a bike or not by defining primitives over bounding box characteristics of the objects in the image. This query results in significant class imabalance in the labeled and unlabeled dataset, with the labeled dataset containing only 18% positive examples. *We use this dataset to measure how Reef performs with a small number of easily interpretable primitives and with a skewed labeled dataset to learn heuristics based on.*

MS-COCO [27] In this multi-modal dataset, each image has five text caption associated with it and the goal was to build a classifier to determine whether there was a person in the image. Reef only has access to the bag-of-words representations of these captions as primitives and the deep learning end model has access only to the images associated with the captions. *We use this dataset to demonstrate how Reef can operate over the text domain and how the domain that heuristics are generated for can be different from the domain the final classifier is running on.*

IMDb The task was to classify plot summaries of movies as describing action or romance movies. Users had previously developed a collection of 6 heuristics to label this data; however, we found that Reef was able to outperform these hand-written heuristics for a task that did not require any domain expertise. *We use this dataset to demonstrate how Reef can outperform user-based heuristics for text-based tasks that do not require any domain expertise.*

Twitter [28] The originally crowdsourced task is to determine whether a specific tweet has a positive or negative connotation. Since less than 1% of the data was labeled by crowdworkers, Reef used that subset as the labeled set. With such a significant difference between the labeled and unlabeled dataset, we wanted to see how well Reef can generalize. Moreover, results of this task has interesting follow on work for how to make crowdsourcing more efficient. *We use this dataset to demonstrate how Reef can work even when the labeled dataset is signficantly smaller than the unlabeled set and how this can help make crowdsourcing more efficient.*

Chemical-Disease Relation Extraction (CDR) [53] The task is to detect relations among chemicals and diseases mentioned in PubMed abstracts [53]. Previously, a combination of distant supervision from the Comparative Toxicogenomics Database [9] and user-defined heuristics were used to build a training set for this task. Unlike previous datasets, the heuristics contain additional, structured information from the knowledge base, which puts Reef at a disadvantage. *We use this dataset to demonstrate how Reef generated heuristics compare to these complex rules and* external sources of information.

B. PROOF OF PROPOSITION 1

Proposition 1: *Suppose we have M heuristics with empirical accuracies $\hat{\alpha}$, accuracies learned by the generative model $\tilde{\alpha}$, and measured error $||\hat{\alpha} - \tilde{\alpha}||_\infty \leq \epsilon$ for all M iterations. Then, if each heuristic labels a minimum of*

$$N \geq \frac{1}{2(\gamma-\epsilon)^2} \log\left(\frac{2M^2}{\delta}\right)$$

datapoints at each iteration, the generative model will succeed in learning accuracies within $||\alpha^* - \tilde{\alpha}||_\infty < \gamma$ across all iterations with probability $1-\delta$.

Proof: We first start by using the triangle inequality to bound the probability of the l_∞ norm error between the learned and true accuracies being larger than γ, $||\alpha^* - \tilde{\alpha}||_\infty \leq \gamma$.

$$\begin{aligned} P(||\alpha^* - \tilde{\alpha}||_\infty > \gamma) &\leq P(||\alpha^* - \hat{\alpha}||_\infty + ||\hat{\alpha} - \tilde{\alpha}||_\infty > \gamma) \\ &\leq P(||\alpha^* - \hat{\alpha}||_\infty + \epsilon > \gamma) \end{aligned} \tag{2}$$

where ϵ is the l_∞ norm error between the learned and empirical accuracies.

We then bound the probability of the l_∞ norm error between the empirical and true accuracies being greater than γ. By the union bound,

$$P(||\alpha^* - \hat{\alpha}||_\infty + \epsilon > \gamma) \leq \sum_{i=1}^{M} P(||\alpha_i^* - \hat{\alpha}_i|| + \epsilon > \gamma) \tag{3}$$

We rewrite $\hat{\alpha}_i$ for heuristics $i = 1, \ldots, M$ as

$$\hat{\alpha}_i = \frac{1}{N_i} \sum_{j: y_j \in \{-1,1\}} \mathbb{1}(\hat{y}_{ij} = y_j^*)$$

where $\hat{y}_{ij}$ is the label heuristic i assigned to datapoint j, y_j^* is the true label for datapoint j, and N_i is the number of datapoints where $\hat{y}_{ij} \in \{1, -1\}$ and did not abstain.

Substituting the above expression in (3), we get

$$\begin{aligned} &P\left(||\alpha_i^* - \hat{\alpha}_i||_\infty + \epsilon > \gamma\right) = P\left(||\alpha_i^* - \hat{\alpha}_i||_\infty > \gamma - \epsilon\right) \\ &\leq \sum_{i=1}^{M} P\left(|\alpha_i^* - \frac{1}{N_i} \sum_{j=1, y_j \in \{-1,1\}} \mathbb{1}(\hat{y}_{ij} = y_j^*)| > \gamma - \epsilon\right) \\ &\leq \sum_{i=1}^{M} 2\exp\left(-2(\gamma-\epsilon)^2 N_i\right) \\ &\leq 2M \exp\left(-2(\gamma-\epsilon)^2 \min(N_1, \ldots, N_M)\right) \end{aligned} \tag{4}$$

where the second step uses Hoeffding's inequality and assumes that the datapoints are independent.

The above expression bounds the probability of the generative model failing *per iteration* of Reef. To bound the probability of failure across all iterations, we use the union bound:

$$\begin{aligned} &P(||\alpha_i^* - \hat{\alpha}_i||_\infty > \gamma - \epsilon \text{ for any iteration}) \\ &\qquad \leq \sum_{i=1}^{M} P(||\alpha_i^* - \hat{\alpha}_i||_\infty > \gamma - \epsilon \text{ for one iteration}) \\ &\qquad \leq \sum_{i=1}^{M} p \\ &\qquad = Mp \end{aligned}$$

where $P(||\alpha_i^* - \hat{\alpha}_i||_\infty > \gamma - \epsilon) = p$ is the failure probability for a single iteration.

With the failure probability over all M iterations, $\delta = P(||\alpha_i^* - \hat{\alpha}_i||_\infty > \gamma - \epsilon \text{ for any iteration})$, we can express the failure probability of a single iteration as $p = \frac{\delta}{M}$. Substituting into (4), we get

$$\begin{aligned} \frac{\delta}{M} &= P\left(||\alpha^* - \tilde{\alpha}||_\infty > \gamma - \epsilon\right) \\ \delta &= MP\left(||\alpha^* - \tilde{\alpha}||_\infty > \gamma - \epsilon\right) \\ &\leq 2M^2 \exp\left(-2(\gamma-\epsilon)^2 \min(N_1, \ldots, N_M)\right) \end{aligned}$$