ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240749

Combining Active Learning and Learning to Reject for Anomaly Detection

Luca Stradiotti [a,b,*], **Lorenzo Perini**[a,b] **and Jesse Davis**[a,b]

[a]Department of Computer Science, KU Leuven, Belgium
[b]Leuven.AI

Abstract. Anomaly detection attempts to identify instances in the data that do not conform to the expected behavior. Because it is often difficult to label instances, the problem is tackled in an unsupervised way by employing data-driven heuristics to identify anomalies. However, the heuristics are imperfect which can degrade a detector's performance. One way to mitigate this problem is using Active Learning to collect labels that help correct cases where the employed heuristics are incorrect. Alternatively, one can allow the detector to abstain (i.e., say "I do not know") whenever it is likely to make mispredictions at test time, which is called Learning to Reject (LtR). However, while both have been studied in the context of anomaly detection, they have not been considered in conjunction. Although they both need labels to accomplish their task, integrating these two ideas is challenging for two reasons. First, their label selection strategies are intertwined but they acquire different types of labels. Second, it is unclear how to best divide the limited budget between labeling instances that help AL and those that help LtR. In this paper, we introduce SADAL, the first semi-supervised detector that allocates the label budget between AL and LtR by relying on a reward-based selection function. Experimentally on 25 datasets, we show that our approach outperforms several baselines by achieving a better performance.

1 Introduction

Anomaly detection attempts to automatically identify instances that do not follow expected patterns [5]. These instances, named anomalies, are usually associated with critical events such as failures in manufacturing [43], breakdowns in wind turbines [39], or water leaks in stores [38]. These critical events usually come with a negative cost such as the monetary costs of lost products or societal costs such as wasting vital resources. The timely detection of anomalies offers the potential to reduce these costs.

Anomaly detection algorithms often operate in an unsupervised manner because labels, especially for the anomalies, are usually hard to collect: anomalies are rare (e.g., you may need to inspect thousands of instances before encountering an anomaly) and infeasible to generate (e.g., voluntarily breaking a machine to collect an anomaly) [22, 29]. Unsupervised anomaly detectors overcome the absence of labels by employing heuristic intuitions, such as that anomalies fall in low-density regions [4], to identify anomalous behavior. Choosing the right heuristic is crucial to having an accurate anomaly detector. Unfortunately, the chosen heuristic often is not appropriate for all types of anomalies in a domain. Consequently, detectors can perform poorly when confronted with anomalous behavior that does not align with the chosen heuristic, which can diminish a user's trust in the detector's predictions [18, 26].

Incorporating a human-in-the-loop can help in two different ways. On the one hand, it is possible to use active learning (AL) [1] to collect labels and move to a semi-supervised setting. AL queries the labels for training instances that fall in the regions where the detector is likely to make mispredictions [9, 30]. By learning from the newly collected labels, a semi-supervised anomaly detector can significantly improve the prediction quality [36]. On the other hand, one can use learning to reject (LtR) [19, 35, 8] where the detector has the option to abstain (i.e., return "I do not know") from making a prediction in situations where it is at a heightened risk of making a mistake. When a detector abstains, the human would need to intervene to make a decision. By understanding its limitations, a detector's performance improves when it does offer a prediction, which increases the user's trust in the model.

AL and LtR both require labeled instances, but they require different types of labels. AL strategically targets labels that will improve the detector, which introduces a strong bias in the set of collected labels. In contrast, LtR needs to assess the detector's performance so it needs an unbiased or weakly-biased sample. The mismatch between the type of labels needed means that AL and LtR are often treated independently. Nonetheless, trying to integrate these approaches introduces two interesting interdependencies. First, their instance selection strategies are dependent on each other in a detector with rejection. The detector would not benefit much from labeling instances within the rejection region, as it would not make predictions for similar test instances. Similarly, learning the rejection region (i.e., the region of the space where the detector abstains from predicting) depends on the detector's output, which changes according to which training instances are labeled. Second, one often has a single budget to label instances, and deciding how many instances to allocate to each of AL and LtR is challenging. There is an inherent tension between these two objectives. On the one hand, allocating more of the budget to AL will increase the detector's prediction quality while, on the other hand, using more for LtR results in a better estimate of the detector's performance, hence making it possible to determine when abstention is warranted.

In this paper, we introduce SADAL (Semi-supervised Anomaly Detector combining Active learning and Learning to reject) a novel

* Corresponding Author. Email: luca.stradiotti@kuleuven.be

semi-supervised anomaly detector with rejection that collects new labels accounting for the interdependence between active learning and learning to reject. For this task, SADAL starts from an unsupervised setting. In each round, it uses a reward function to decide whether to allocate the budget towards acquiring training labels (i.e., AL) or to determine when the model should abstain (i.e., LtR). For both AL and LtR, we come up with approaches tailored to our setting to select which instances should be labeled. An extensive experimental analysis on 25 benchmark and real-world datasets shows that our approach outperforms its competitors.

Contributions. Our main contributions in this work are:

- we introduce the problem of allocating the label budget between AL and LtR in order to actively train an anomaly detector with a reject option
- we design (i) a novel AL strategy for a semi-supervised anomaly detector with rejection, and (ii) a novel sampling strategy to acquire labels for LtR when the classes are imbalanced;
- we propose a reward function that decides whether to collect the labels for AL or LtR;
- we empirically perform an extensive analysis on 25 real-world and benchmark datasets to compare our proposed method to several baselines.

2 Preliminaries and Related Work

Let X be random variables representing a real d-dimensional feature vector and Y a random variable representing its class label, where 0 denotes a normal instance and 1 an anomaly. Let $\mathcal{D} = \{x_1, x_2, ..., x_m\}$ be an unlabeled dataset with m instances .

Anomaly detection (AD) An anomaly detector $h\colon \mathbb{R}^d \to \mathbb{R}^+$ maps an instance x to an anomaly score $h(x)$ that represents the instance's degree of anomalousness (i.e., a higher score means that an instance is more anomalous). Typically, unsupervised anomaly detectors leverage heuristic intuitions about what behavior is anomalous to assign anomaly scores. However, raw scores can be hard for a human to interpret [26]. Thus, existing approaches convert the scores into a class conditional probability $\mathbb{P}(Y = 1|X = x)$ by employing squashing functions $\mathcal{S}$ [38]:

$$p_x \coloneqq \mathbb{P}(Y = 1|X = x) = \mathcal{S}_\lambda(h(x)) = 1 - 2^{-\frac{h(x)^2}{\lambda^2}}, \quad (1)$$

where λ is the decision threshold, which is often set in function of the dataset's contamination factor γ (i.e., the expected proportion of anomalies) [28, 27, 25]. Strictly speaking, $\mathcal{S}_\lambda$ is a monotonic function that assigns probabilities ≥ 0.5 to anomaly scores $\geq \lambda$. Thus, a detector's prediction can be obtained from p_x as

$$f(x) = \begin{cases} 0 & p_x < 0.5; \\ 1 & p_x \geq 0.5. \end{cases} \quad (2)$$

Active Learning (AL) Unsupervised anomaly detectors tend to make mispredictions for instances close to the decision boundary, as the lack of labels does not allow a detector to properly distinguish between normals and anomalies. Existing work has shown that acquiring training labels for those instances yields an improvement in a detector's performance, when retrained on the enriched training data [40]. Thus, given a label budget B, Active Learning is a set of strategies that aim to select the instances where the detector is likely to make incorrect predictions and query their label to the user. Commonly, pool-based Active Learning (AL) strategies iteratively query $b \leq B$ instances before retraining the model [23, 21, 9]. The most common AL strategy is Uncertainty Sampling [17], which selects the instances in the training set $\mathcal{T}$ that are closer to the decision boundary by measuring the margin between the two class conditional probabilities as:

$$x_t = \arg\min_{x_i \in \mathcal{T}} \{|p_{x_i} - (1 - p_{x_i})|\} = \arg\min_{x_i \in \mathcal{T}} \{|p_{x_i} - 0.5|\}.$$

Other existing AL strategies propose to select instances exploiting different heuristics, like diversity and anomalousness [1, 10, 11, 13, 33, 2] but are less common in anomaly detection.

Learning to Reject (LtR) Alternatively, one can reduce the detector's mistakes by limiting the number of predictions. That is, Learning to Reject (LtR) allows a model to abstain (i.e., reject) from making a prediction for a test instance whenever the detector is likely to be incorrect [19]. For this, we include a detector-agnostic reject option that acts as a filter based on the detector's output [42, 14, 34, 7]. Formally, learning a classifier with rejection entails (1) estimating the model's confidence in predictions c_x and (2) learning a rejection threshold $\tau \in [0, 1]$ such that predictions with low confidence (equivalent to high probability of misprediction) are rejected, i.e., deferred to the user. A common confidence function measures the margin between the two class conditional probabilities [24] as

$$c_x = |\mathbb{P}(Y = 1|X = x) - \mathbb{P}(Y = 0|X = x)| = |2p_x - 1|. \quad (3)$$

A standard approach for setting the rejection threshold τ on c_x is to evaluate the model on a labeled validation set $\mathcal{V}$ for different threshold values [16, 31, 14]: on one hand, a low τ increases the chance of mispredictions while, on the other hand, a high τ decreases the model's usage (i.e., forces the human to intervene). Optimizing this trade-off often requires defining the cost of mispredictions ($C_{FP} > 0$ for false positives, $C_{FN} > 0$ for false negatives) and rejections ($C_R > 0$).

However, Tortorella [37] showed that thresholding the confidence differently per predicted class improves the model's performance, especially when dealing with limited labeled instances. Denoting by τ_n and τ_a the rejection thresholds for the normal and anomaly class, finding their value entails optimizing the cost functions:

$$\tau_n = \arg\min_{0<t_n<1} \left[\mathbb{P}(c_x \leq t_n|y=0)C_R + \mathbb{P}(c_x > t_n, p_x < 0.5|y=1)C_{FN}\right]$$
$$\tau_a = \arg\min_{0<t_a<1} \left[\mathbb{P}(c_x \leq t_a|y=1)C_R + \mathbb{P}(c_x > t_a, p_x \geq 0.5|y=0)C_{FP}\right]$$

At test time, an instance x is rejected if either (1) $p_x \geq 0.5$ and $c_x \leq \tau_a$ or (2) $p_x < 0.5$ and $c_x \leq \tau_n$. Because the confidence is normally obtained as a function of the class conditional probabilities (Eq. 3), one can equivalently set the rejection thresholds over p_x by transforming τ_n and τ_a into $\tau_N = \frac{1+\tau_n}{2}$ and $\tau_A = \frac{1-\tau_a}{2}$. From now on, we use τ_N, τ_A to refer to the rejection thresholds.

3 Methodology

Our goal is to learn an anomaly detector with rejection $f_{®}\colon \mathbb{R}^d \to \{0, 1, ®\}$ of the form:

$$f_{®}(x) = \begin{cases} 0 & \text{if } p_x < \tau_N; \\ ® & \text{if } p_x \in [\tau_N; \tau_A]; \\ 1 & \text{if } p_x > \tau_A. \end{cases}$$

We will approach this problem from a semi-supervised perspective. An anomaly detector with rejection can benefit from acquiring labels in two different ways. On the one hand, collecting more labels via AL to train the detector will improve its performance. On the other hand, having more labels will allow setting more accurate rejection thresholds via LtR.

Unfortunately, there is a single, limited budget that must be used for both tasks. This poses two challenges. First, AL and LtR are strongly intertwined: allocating labels to one affects the other. That is, allocating the budget to improve the detector will result in a better calibration of the detector's probabilities, which simplifies learning the rejection thresholds, to the extent that, with perfectly calibrated probabilities, the optimization problem narrows down to setting a constant value. Similarly, allocating the budget to LtR enables setting better rejection thresholds. This affects the AL's selection strategy, to the extent that, with optimal thresholds, AL would only target instances outside the rejection region. A natural question is how to design label selection strategies that account for this interdependence. Second, AL and LtR need different types of labels to achieve their objectives: AL targets the regions where the detector performs poorly, thus introducing a strong bias in the labeled population, while LtR requires a reliable estimate of the detector's performance over the entire instance space, thus preferring un/weakly-biased labeled instances. This introduces the challenge that in each round, we must decide whether to acquire labels that help improve the detector's performance (AL) or to better set the rejection thresholds (LtR). It is unlikely that evenly dividing the budget between each objective will yield optimal performances (see the experiments).

We tackle these challenges by introducing SADAL, a novel detector with rejection utilizing a K-round allocation loop considering the interplay between Active Learning and Learning to Reject. Initially, SADAL partitions the unlabeled dataset $\mathcal{D}$ into the training set $\mathcal{T}$ to learn the detector and the validation set $\mathcal{V}$ to optimize the rejection thresholds. The detector h_0 is trained with no labels, the rejection thresholds τ_A and τ_N are set to default values, and the budget B is split into K allocation rounds, where $b = \lfloor B/K \rfloor$ labels are acquired. In the first two rounds, SADAL collects b validation and b training labels, and measures the initial rewards $\mathcal{R}^e_{\mathcal{V}}(1)$ and $\mathcal{R}^e_{\mathcal{T}}(2)$, i.e., how beneficial the new training and validation labels are for the model. The allocation loop begins, distributing $B - 2b$ of the remaining budget over $K - 2$ rounds. Labels are obtained from either the training or validation set based on the option yielding the highest reward. After a training set allocation, the detector is retrained with new labels; after a validation set allocation, the rejection thresholds are optimized again. Then, SADAL updates the rewards for the next allocation round by measuring the impact of the past labels. Figure 1 shows how SADAL's performance improves when more labels are added (center) to $\mathcal{T}$ and (right) to $\mathcal{V}$ on a 2D toy test set. See Algorithm 1 for the model's pseudo-code. Next, we discuss the two key points of our approach, namely how to select the labels to train the detector and to optimize the rejection thresholds (Sec. 3.1) and how to measure the reward for the next allocation round (Sec. 3.2).

Algorithm 1 SADAL

Given : $\mathcal{D}$ - unlabeled dataset with contamination factor γ, h - anomaly detector, B - overall budget, K - number of rounds

split randomly $\mathcal{D}$ into $\mathcal{T}$ and $\mathcal{V}$
train h_0 unsupervised and set $\tau_N = 0.5 - \frac{\gamma}{2}$, $\tau_A = 0.5 + \frac{\gamma}{2}$
collect b labels from $\mathcal{V}$ as in Sec. 3.1, update τ_N and τ_A
measure $\mathcal{R}^e_{\mathcal{V}}(1)$ as in Sec. 3.2
collect b labels from $\mathcal{T}$ as in Sec. 3.1, update h
measure $\mathcal{R}^e_{\mathcal{T}}(2)$ as in Sec. 3.2
$b = B/K$
$k = 3$
while $k \leq K$ **do**
  if $\mathcal{R}^e_{\mathcal{T}}(k-1) > \mathcal{R}^e_{\mathcal{V}}(k-1)$ **then**
    collect b labels from $\mathcal{T}$ as in Sec. 3.1 and update h
    measure $\mathcal{R}^e_{\mathcal{T}}(k)$ and $\mathcal{R}^e_{\mathcal{V}}(k)$ as in Sec. 3.2
    update τ_A and τ_N
  else
    collect b labels from $\mathcal{V}$ as in Sec. 3.1 and update τ_N and τ_A
    measure $\mathcal{R}^e_{\mathcal{T}}(k)$ and $\mathcal{R}^e_{\mathcal{V}}(k)$ as in Sec. 3.2
  end if
  $k = k + 1$
end while

3.1 How to select which instances to label

Selecting strategic instances to enhance a detector with rejection requires accounting for two important aspects. First, labeling training instances that fall in the rejection region may not affect the detector's performance, as a similar test instance would be rejected. Thus, such labels might be wasted. Second, selecting i.i.d. instances to optimize the rejection thresholds is likely to end up labeling only normal instances due to high class imbalance. Thus, learning the rejection thresholds might become an ill-posed optimization problem.

a) Collecting labels to train the detector Existing AL approaches tend to acquire labels for instances that are close to the decision boundary [40]. However, for a model with rejection, such instances lie in the rejection region. Acquiring labels in the rejection region will not improve a detector's performance at test time because the model will abstain from making a prediction. Our intuition is that in a rejection setting, a detector's decision boundary lies around the rejection thresholds. Consequently, we propose a novel rejection-aware Active Learning strategy that queries the b unlabeled instances in $\mathcal{T}$ that are closest to τ_N and τ_A and fall outside of the rejection region:

$$x_t = \underset{x_i \in \mathcal{T}}{\arg\min} \{\min(|p_{x_i} - \tau_N|, |p_{x_i} - \tau_A|)\}\,, \text{ s.t. } p_{x_i} \notin [\tau_N; \tau_A].$$

b) Collecting labels to optimize the rejection thresholds. To estimate the rejection thresholds we need validation labels that satisfy two criteria. On the one hand, they should give an accurate estimate of a detector's performance. This suggests a (1) i.i.d. strategy would be appropriate. On the other hand, it is important to identify those regions of the instance space where the model performs worse (i.e., is more likely to make a misprediction). This suggests a (2) biased sampling strategy. We trade off these two criteria by proposing a weakly-biased strategy that samples the instances uniformly over their class conditional probability values, rather than over the data density. Strictly speaking, we (1) sample $p^* \sim \text{UNIF}(0, 1)$, and (2) query the instance whose conditional probability p_x is closer to p^*:

$$x_v = \underset{x_i \in \mathcal{V}}{\arg\min} \{|p_{x_i} - p^*|\}\,, \text{ where } p^* \sim \text{UNIF}(0, 1).$$

This weakly-biased sampling mechanism targets instances from both classes equally, overcoming the class imbalance issue.

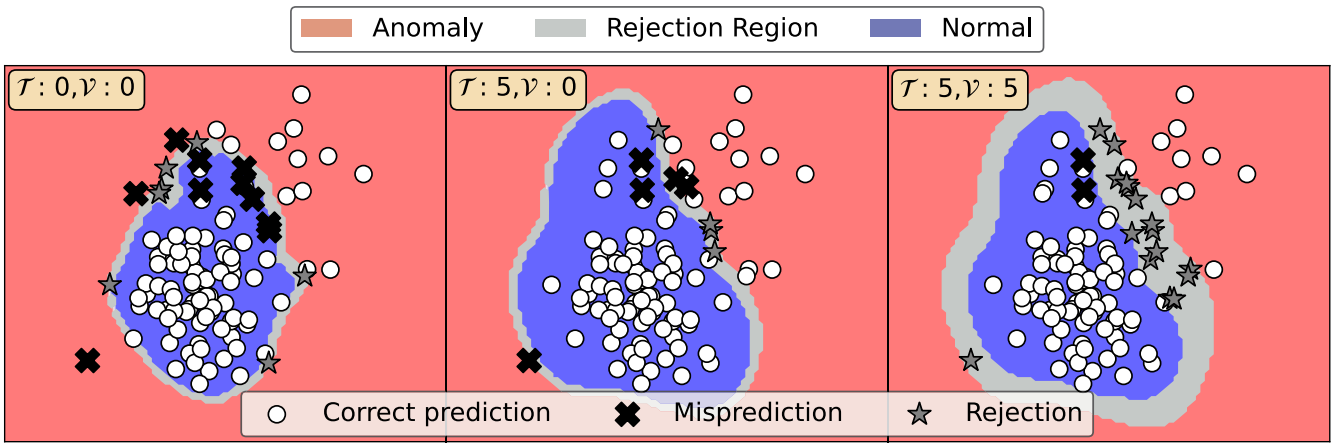

Figure 1: Illustration of how collecting labels via SADAL improves the performance of DEEPSAD on a 2D toy *test set*. The boxes on the top indicate the number of available labels, while the background colors show the prediction (red for anomalies, blue for normals) and the rejection region (gray). SADAL starts with no labels available (left), then collects five labels from $\mathcal{T}$ (center), and five labels from $\mathcal{V}$ (right). Both labeling $\mathcal{T}$ and $\mathcal{V}$ reduce the number of mispredictions (black cross marks): the former enhances the quality of DEEPSAD's output, while the latter limits the number of (incorrect) predictions.

3.2 *Reward function for budget allocation*

Ideally, at each round, one would allocate the budget to the option that results in the best performance after all allocation rounds. However, future knowledge is not available during training/validation. Thus, we propose to look at the past rounds to measure the reward of acquiring new labels. Our insight is that a *larger variation in the detector's output implies a larger gain in performance at test time*. Following this, we design two reward functions for the instances to train the detector and to optimize the rejection thresholds.

Reward function for the labels in $\mathcal{T}$. We look at how the class conditional probabilities vary when retraining the detector with the newly acquired training labels, as a large variation implies that the labels had a strong impact. For this task, we compute the expectation of the absolute difference between the entropy of p_x before and after retraining the detector. Moreover, one wants to average over all historical changes resulting from the acquired training labels, assigning a lower weight to such variations the earlier in the rounds they occurred.

For this task, we include an exponential decay. As a result, the reward for the k-th allocation round is computed as:

$$\mathcal{R}^e_{\mathcal{T}}(k) = \sum_{i=1}^{k} \alpha^{\phi_{\mathcal{T}}(i,k)} \cdot \mathbb{E}_{x\in\mathcal{D}}\left[\left|\mathcal{H}_{h_i}(x) - \mathcal{H}_{h_{i-1}}(x)\right|\right] \quad (4)$$

$$\mathcal{H}_h(x) = -p_x \log_2 p_x - (1-p_x)\log_2(1-p_x).$$

where h_i is the detector learned using the acquired labels up to the i-th round (h_0 is unsupervised), $\phi_{\mathcal{T}} : \mathbb{N}^2 \to \mathbb{N}$ returns for how many rounds labels were collected from $\mathcal{T}$ between the i-th and the k-th round, and α is a hyperparameter to weigh the influence of the past allocation rounds. Note that if the budget is not allocated to the training set for a round $j \leq k$, then the j-th round does not contribute to the k-th reward because $\mathcal{H}_{h_j}(x) = \mathcal{H}_{h_{j-1}}(x)$ for any x.

Reward function for the labels in $\mathcal{V}$. We proceed similarly to the reward for the labels in $\mathcal{T}$ but look at the rejection probabilities instead of the class conditional probabilities. Specifically, the reward in this case is computed as in Equation 4 by substituting $\phi_{\mathcal{T}}$ with $\phi_{\mathcal{V}}$, and $\mathcal{H}_h$ with $\mathcal{H}_{\textcircled{R}}$, which is equal to

$$\mathcal{H}_{\textcircled{R}}(x) = -p^{\textcircled{R}}_x \log_2 p^{\textcircled{R}}_x - \left(1-p^{\textcircled{R}}_x\right)\log_2\left(1-p^{\textcircled{R}}_x\right),$$

where $p^{\textcircled{R}}_x$ represents the rejection probability for an instance x, i.e., the likelihood that x will be rejected if seen at test time.

Estimating the rejection probabilities is not straightforward. Roughly speaking, rejected instances should have a rejection probability ≥ 0.5, while accepted instances should have $p^{\textcircled{R}}_x < 0.5$. Having access to p_x, we need to map the class conditional probabilities to rejection probabilities. Because the rejection thresholds define the boundaries of the rejection region, a desired property is that $p^{\textcircled{R}}_{\tau_N} = p^{\textcircled{R}}_{\tau_A} = 0.5$. Thus, we propose to estimate the rejection probabilities by leveraging the squashing function $\mathcal{S}$ [38] with the two rejection thresholds playing the role of the decision threshold in Equation 1:

$$p^{\textcircled{R}}_x = \begin{cases} \mathcal{S}_{\tau_N}(p_x) & \text{if } p_x < 0.5; \\ \mathcal{S}_{1-\tau_A}(1-p_x) & \text{if } p_x \geq 0.5. \end{cases}$$

4 Experiments

Empirically, we answer the following research questions:

Q1: Does SADAL outperform the baselines?
Q2: How does SADAL behave when varying C_{FP}, C_{FN}?
Q3: How does each component of SADAL impact its performance?

Additionally, the online supplement[1] evaluates how SADAL performs when some annotated anomalies are available from the beginning.

4.1 *Experimental Setup*

Methods. Given that no existing methods directly address our problem, we consider five possible strategies one may adopt in this setting and compare SADAL[2] against the following baselines:

[1] https://github.com/ML-KULeuven/SADAL/blob/main/supplement.pdf
[2] Code available at: https://github.com/ML-KULeuven/SADAL.

- ALLINAL assigns the full budget to AL to improve the detector's performance. The same labels used to train the detector are used to set the rejection thresholds;
- ALLINLTR learns an unsupervised detector and assigns the full budget to optimize the rejection thresholds via LtR. The detector is trained in a fully unsupervised manner;
- HALF splits equally the budget between AL and LtR; First, the detector collects half of the labels via AL in multiple rounds. Then the rest of the budget is allocated to obtain labels to optimize the rejection thresholds.
- RANDOM randomly selects at each round between assigning the budget to AL or LtR.
- ONLYAL does not include any reject option. The full budget is assigned to AL to improve the detector's performance.

For AL, we always use Uncertainty Sampling [17] to select the instances to label. While, when the budget is allocated to LtR, the instances are randomly selected. None of the baselines use our reward function because it is one of our main contributions.

Data. We run our experimental analysis on two real-world wind turbine datasets and the 23 benchmark datasets described in Table 1 (Supplement) that are widely used in anomaly detection [18]. The task for the two turbine datasets (T15 and T21) is to detect blade icing (i.e., the anomalies), which could potentially damage the turbines and slow power production [41]. Various measurements (e.g., wind speed and power) are collected every seven seconds for either two months (T15) or one month (T21). Following [41], we construct feature-vectors by averaging over segments of 1 minute.

Evaluation metric. In learning with rejection, abstaining often improves the performance on the instances for which the model makes a prediction. However, abstaining has a cost because it pushes work to a human. Therefore, often one takes a cost-based approach where both mispredictions and rejections have an associated cost [8, 24, 3, 6]. We use the following metric:

$$\mathcal{C}_f = C_R \cdot \mathbb{P}\left(f_{®}(X) = ®\right) + C_{FP} \cdot \mathbb{P}\left(f_{®}(X) = 1 \mid Y = 0\right) + C_{FN} \cdot \mathbb{P}\left(f_{®}(X) = 0 \mid Y = 1\right) \tag{5}$$

where $C_R, C_{FP}, C_{FN} > 0$ are the costs of a rejection, a false positive, and a false negative respectively. We assume correct predictions have a cost of zero, while we set $C_{FP} = C_{FN} = 1$. We consider rejection less costly than misprediction and the rejection cost needs to satisfy the inequality $C_R \leq min\{C_{FP} \times (1-\gamma), C_{FN} \times \gamma\}$, otherwise one could predict always normal and pay an expected cost of $C_{FN} \times \gamma$ or anomaly and pay $C_{FP} \times (1-\gamma)$. We always set $C_R = min\{C_{FP} \times (1-\gamma), C_{FN} \times \gamma\}$.

Setup. For each dataset, we employ the following procedure: (i) we split the dataset into training, validation, and test set ($40\% - 40\% - 20\%$), (ii) we fit the anomaly detector on the unlabeled training set and set two default rejection thresholds to $0.5 \pm \frac{\gamma}{2}$; (ii) we collect b validation labels and b training labels; (iii) we optimize the rejection thresholds and measure the initial validation labels' reward; (iv) we train the anomaly detector on the partially labeled training set and measure the initial training labels' reward; (v) we allocate the next round budget b to the option with the highest reward and repeat (iii) or (iv) for $K-2$ allocation rounds. During each of the steps, we measure the detector's performance on the test set using the cost function in Equation 5. We set B to be 30% of the combined size of the training and validation sets, and $b = 2\%$. Consequently, we run 15 allocation rounds. To obtain more consistent results, we repeat (i - v) 10 times and report the average results. Therefore, we run in total $25 \times 15 \times 10 = 3000$ experiments.

Hyperparameters We use DeepSAD [32] with its default hyperparameters as the semi-supervised anomaly detector. For LtR, we set the two rejection thresholds through Bayesian Optimization (GP MINIMIZE implemented in SKOPT) with 30 calls [15] and limit the rejection rate on the validation set to be at most 50% (this threshold prevents the model from rejecting all the instances). The hyperparameter α that weights the influence of the previous allocation rounds on the actual reward is set to 0.5.

4.2 Results

Q1: models' comparison. Figure 2 shows a fine-grained view of the results by plotting the average test cost per instance C_f as a function of the allocation round k. On average, SADAL outperforms all baselines by reducing the test cost by approximately 10% vs HALF and RANDOM, 11% vs ALLINLTR, 13% vs ALLINAL, and 50% vs ONLYAL. Moreover, SADAL achieves lower/similar (i.e., differences ≤ 0.001) test cost in around 78% of the experiments against the two runner-ups HALF and RANDOM.

For each experiment, we rank the methods from the best (rank 1) to the worst (rank 6) and report the average ranks in Table 1. Results show that SADAL consistently obtains the lowest (best) average rank when aggregating for each allocation round over all datasets. Remarkably, all the baselines that include a reject option achieve similar average positions (around 3), indicating that their performance strictly depends on the allocated budget and dataset.

Finally, we analyze the statistical significance of the models' performance separately for each allocation round k. For any k, the Friedman test [12] rejects the null hypothesis that all the methods perform similarly (p-value $< 10^{-5}$). Moreover, the Nemenyi post-hoc statistical test with $\alpha = 0.05$ finds that SADAL is significantly better than all baselines in the vast majority of allocation rounds, with the only exceptions for ALLINLTR/ALLINAL at $k = 1$, and for HALF at $k = 11, 15$.

Q2: varying C_{FP} and C_{FN}. The misprediction costs C_{FP} and C_{FN} usually depend on the application domain: in some contexts, one must avoid false alarms, while in others, undetected anomalies may be harmful. In this experiment, we vary the costs to study how SADAL works in 4 representative settings: (i) C_{FP} is higher than C_{FN} ($C_{FP} = 5$, $C_{FN} = 1$), (ii) C_{FP} is much higher than C_{FN} ($C_{FP} = 20$, $C_{FN} = 1$), (iii) C_{FP} is lower than C_{FN} ($C_{FP} = 1$, $C_{FN} = 5$), and (iv) C_{FP} is much lower than C_{FN} ($C_{FP} = 1$, $C_{FN} = 20$). We always set the rejection cost $c_R = min\{C_{FP} \times (1-\gamma), C_{FN} \times \gamma\}$ to satisfy the constraint on the costs.

Table 2 shows each method's average cost per instance over all datasets and allocation rounds. We observe that SADAL outperforms the baselines when $C_{FP} > C_{FN}$: it decreases the cost by at least 22% when compared to the runner-up ALLINLTR. Moreover, for both (i) and (ii), at the end of the allocation loop ($k = 15$), SADAL obtains on average a lower test cost on 19, 21, 21, 22, 25 datasets when compared to, respectively, ALLINLTR, ALLINAL, HALF, RANDOM and ONLYAL. This shows that SADAL is very useful when false alarms are costly. When $C_{FN} > C_{FP}$, SADAL still outperforms the baselines, despite achieving a smaller improvement. When $C_{FN} >> C_{FP}$, most of the methods perform similarly

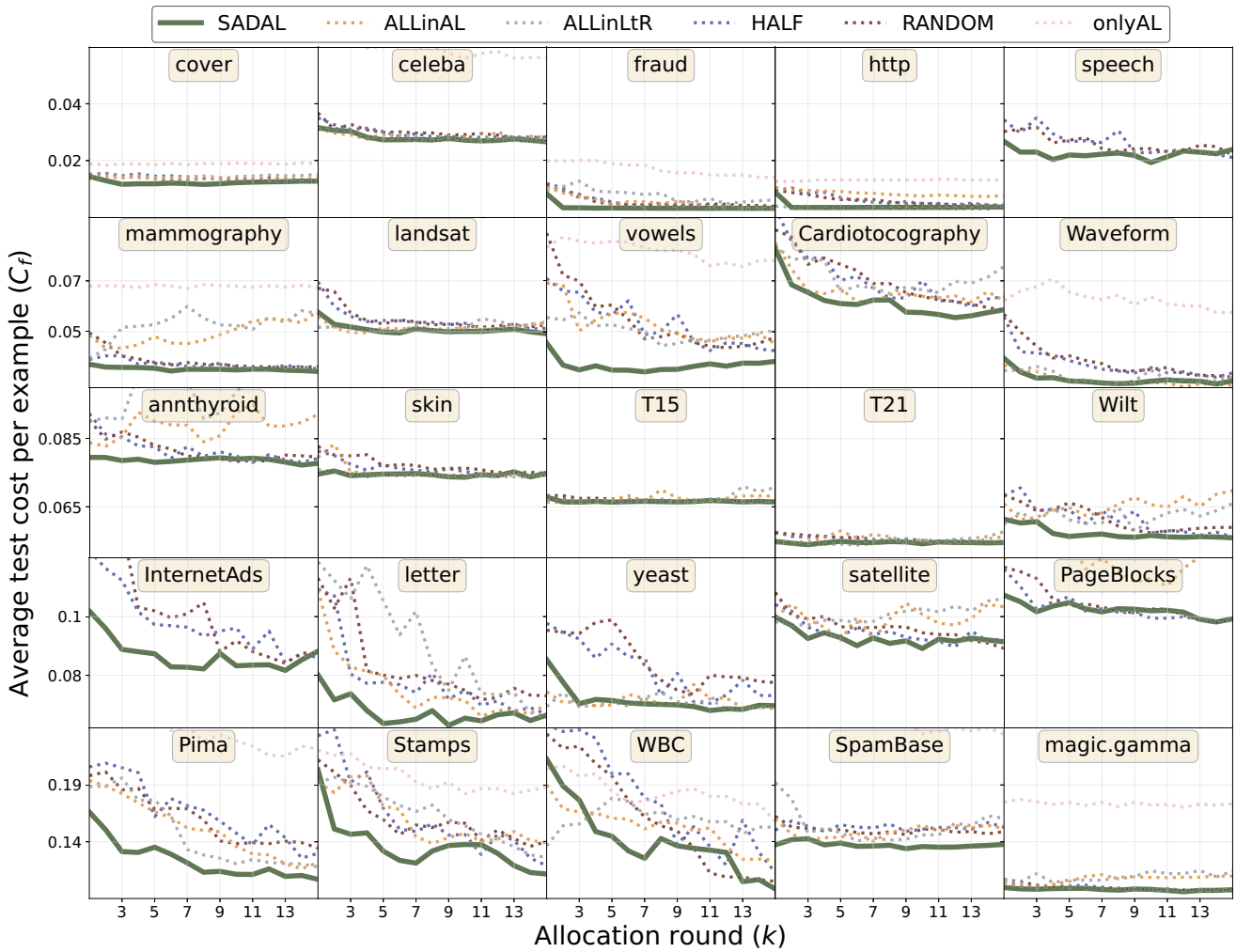

Figure 2: Average test cost per instance for all the considered strategies on all the 25 datasets for 15 allocation rounds (k). Overall, SADAL outperforms the considered baselines on 13 datasets and is competitive in almost all the others.

Table 1: Average rank ($\pm$ std.) for each method across all datasets for 15 allocation rounds. Overall, SADAL outperforms the competing baselines and always achieves the lowest (best) average rank, indicating that it consistently obtains good performances over the experiments.

	Ranks (avg. $\pm$ std.)					
k	SADAL	ALLINAL	ALLINLTR	HALF	RANDOM	ONLYAL
1	**1.84 $\pm$ 0.99**	2.68 $\pm$ 0.99	2.44 $\pm$ 1.58	4.00 $\pm$ 1.22	4.24 $\pm$ 0.88	5.80 $\pm$ 0.65
2	**1.40 $\pm$ 0.71**	2.72 $\pm$ 1.31	3.12 $\pm$ 1.42	3.84 $\pm$ 1.18	4.08 $\pm$ 0.91	5.84 $\pm$ 0.47
3	**1.44 $\pm$ 0.71**	2.76 $\pm$ 1.20	3.24 $\pm$ 1.45	3.68 $\pm$ 1.18	3.96 $\pm$ 1.10	5.92 $\pm$ 0.40
4	**1.36 $\pm$ 0.64**	2.96 $\pm$ 1.14	3.44 $\pm$ 1.47	3.52 $\pm$ 1.36	3.80 $\pm$ 1.04	5.92 $\pm$ 0.40
5	**1.44 $\pm$ 0.71**	3.08 $\pm$ 1.26	3.48 $\pm$ 1.50	3.44 $\pm$ 1.26	3.64 $\pm$ 1.19	5.92 $\pm$ 0.40
6	**1.28 $\pm$ 0.54**	3.40 $\pm$ 1.22	3.40 $\pm$ 1.41	3.36 $\pm$ 1.32	3.68 $\pm$ 1.11	5.88 $\pm$ 0.60
7	**1.40 $\pm$ 0.76**	3.12 $\pm$ 1.42	3.48 $\pm$ 1.36	3.48 $\pm$ 1.23	3.56 $\pm$ 1.08	5.96 $\pm$ 0.20
8	**1.32 $\pm$ 0.63**	3.40 $\pm$ 1.32	3.76 $\pm$ 1.36	3.32 $\pm$ 1.11	3.20 $\pm$ 1.15	6.00 $\pm$ 0.00
9	**1.44 $\pm$ 0.71**	3.56 $\pm$ 1.29	3.56 $\pm$ 1.39	3.16 $\pm$ 1.25	3.28 $\pm$ 1.21	6.00 $\pm$ 0.00
10	**1.28 $\pm$ 0.54**	3.76 $\pm$ 1.01	3.64 $\pm$ 1.47	3.16 $\pm$ 1.25	3.16 $\pm$ 1.11	6.00 $\pm$ 0.00
11	**1.72 $\pm$ 0.84**	3.36 $\pm$ 1.22	3.84 $\pm$ 1.55	2.64 $\pm$ 1.25	3.44 $\pm$ 1.16	6.00 $\pm$ 0.00
12	**1.44 $\pm$ 0.65**	3.52 $\pm$ 1.08	3.96 $\pm$ 1.59	3.00 $\pm$ 1.04	3.08 $\pm$ 1.19	6.00 $\pm$ 0.00
13	**1.52 $\pm$ 0.96**	3.44 $\pm$ 1.16	3.84 $\pm$ 1.52	3.16 $\pm$ 1.14	3.04 $\pm$ 1.14	6.00 $\pm$ 0.00
14	**1.52 $\pm$ 0.65**	3.56 $\pm$ 1.29	3.96 $\pm$ 1.43	2.96 $\pm$ 1.06	3.00 $\pm$ 1.26	6.00 $\pm$ 0.00
15	**1.76 $\pm$ 0.88**	3.56 $\pm$ 1.12	4.00 $\pm$ 1.44	2.60 $\pm$ 1.08	3.08 $\pm$ 1.41	6.00 $\pm$ 0.00

and obtain high costs: this is due to the underlying detector producing too many false negatives that LtR is unable to reject. This is further supported by a quick analysis of the rejection rates. ALLINLTR rejects on average 7% more test instances compared to SADAL: ALLINLTR achieves a lower test cost by optimizing the rejection thresholds more effectively, leveraging the larger number of labels for LtR.

Q3: ablation study. To assess the impact of the three main contributions of SADAL, we create three variants of our method. Each

Table 2: Each method's average test cost per instance averaged over all datasets and allocation rounds. Overall, SADAL outperforms the competing baselines when $C_{FP} > C_{FN}$, while being less effective in the other cases because of its sampling strategy for the LtR labels.

		Average cost per instance C_f					
C_{FP}	C_{FN}	SADAL	ALLINAL	ALLINLTR	HALF	RANDOM	ONLYAL
5	1	**0.122**	0.210	0.161	0.166	0.164	0.495
20	1	**0.343**	0.671	0.488	0.510	0.504	1.833
1	5	**0.290**	0.299	0.299	0.297	0.298	0.336
1	20	0.913	0.953	**0.890**	0.910	0.910	1.077

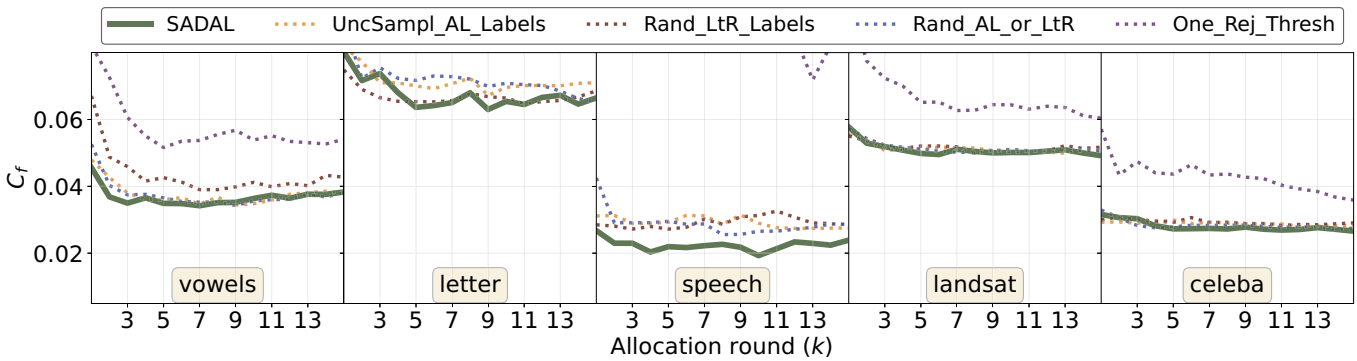

Figure 3: The impact of SADAL's contributions and the two-thresholds plug-in on the average test cost per instance as a function of the allocation round (k) for five representative datasets. All components have a positive impact on the final performance and decrease SADAL's average test cost per instance.

variant removes one contribution:

1. UNCSAMPL_AL_LABELS selects the instances to be labeled to train the detector using standard Uncertainty Sampling instead of our rejection-aware strategy;
2. RAND_LTR_LABELS selects instances to be labeled uniformly at random for setting the rejection thresholds instead of using our weakly-biased sampling approach;
3. RAND_AL_OR_LTR randomly decides in each allocation whether to collect labels to train the detector or to optimize the rejection thresholds as opposed to using our reward function.

For the sake of completeness, we consider ONE_REJ_THRESH that optimizes a symmetric rejection threshold τ centered in 0.5 (i.e., x is rejected if $p_x \in [0.5 - \tau; 0.5 + \tau]$), and keeps the same sampling strategies and reward functions as SADAL.

Figure 3 shows the comparison between SADAL and its four simplified versions on five representative datasets. Results show that all contributions have a positive impact on the final performance of the detector: on average, RAND_AL_OR_LTR and UNCSAMPL_AL_LABELS increase the test cost per instance by 7%, RAND_LTR_LABELS by more than 8% and ONE_REJ_THRESH by more than 44%. Moreover, SADAL obtains a lower/similar (i.e., differences ≤ 0.001) test cost on 64% of the experiments when compared to RAND_AL_OR_LTR and UNCSAMPL_AL_LABELS, on 66% vs RAND_LTR_LABELS, and 95% vs ONE_REJ_THRESH. Thus, each contribution is relevant to achieve the final performance.

5 Conclusion

The literature on semi-supervised anomaly detection has focused on designing new algorithms to learn from labeled and unlabeled data, but largely ignored some practical challenges. First, a recurring assumption is that acquiring training labels always improves a detector's performance. However, the detector might be unable to make accurate predictions in regions where the true class label is ambiguous (e.g., when the key features to recognize an anomaly are missing). In such regions, a better option is to defer the decision to the user at test time. Second, traditional active learning strategies may not be beneficial for a detector with reject option because labeling instances falling within the rejection region may have no impact at test time. Thus, new AL strategies need to be developed. Third, the literature has typically considered using a label budget for a single task (e.g., active learning). However, an unexplored and challenging problem for practitioners is how to smartly distribute it when dealing with multiple tasks.

In this paper, we proposed SADAL, a novel semi-supervised anomaly detector with rejection that allocates the available budget in multiple rounds to reduce the expected cost at test time. When the budget for a round is allocated to the training set, our novel active learning strategy queries the training instances closest to the rejection thresholds but falling outside the rejection region. When the budget is allocated to the validation set, our novel weakly-biased strategy samples instances across the entire instance space. At each round, the budget is allocated depending on a reward function that looks at the past rounds to estimate the expected gain in performance by measuring the entropy of the class conditional probabilities for the training labels, and of the rejection probabilities for the validation ones. We empirically validate our approach on 25 datasets, showing that it often outperforms the baselines and all its contributions have a positive impact on the final performance. In the future, it would be interesting to extend our approach by leveraging insights from the literature on uncertainty quantification [20]. Only instances with high epistemic uncertainty are actually useful to improve the detector via AL, while instances with high aleatoric uncertainty should be rejected.

Acknowledgements

This research is supported by the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme [LS,JD], an FB Ph.D. fellowship by FWO-Vlaanderen (grant 1166224N) [LP], and KUL Research Fund iBOF/21/075 [JD].

References

[1] N. Abe, B. Zadrozny, and J. Langford. Outlier detection by active learning. In *KDD*, pages 504–509. ACM, 2006.

[2] J. T. Ash, C. Zhang, A. Krishnamurthy, J. Langford, and A. Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *ICLR*. OpenReview.net, 2020.

[3] P. L. Bartlett and M. H. Wegkamp. Classification with a reject option using a hinge loss. *J. Mach. Learn. Res.*, 9:1823–1840, 2008.

[4] M. M. Breunig, H. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. In *SIGMOD Conference*, pages 93–104. ACM, 2000.

[5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, 2009.

[6] N. Charoenphakdee, Z. Cui, Y. Zhang, and M. Sugiyama. Classification with rejection based on cost-sensitive classification. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 1507–1517. PMLR, 2021.

[7] C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Trans. Inf. Theory*, 16(1):41–46, 1970.

[8] C. Cortes, G. DeSalvo, and M. Mohri. Learning with rejection. In *ALT*, volume 9925 of *Lecture Notes in Computer Science*, pages 67–82, 2016.

[9] A. Culotta and A. McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, pages 746–751. AAAI Press / The MIT Press, 2005.

[10] I. Dagan and S. P. Engelson. Committee-based sampling for training probabilistic classifiers. In *ICML*, pages 150–157. Morgan Kaufmann, 1995.

[11] S. Dasgupta and D. J. Hsu. Hierarchical sampling for active learning. In *ICML*, volume 307 of *ACM International Conference Proceeding Series*, pages 208–215. ACM, 2008.

[12] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

[13] S. Ebert, M. Fritz, and B. Schiele. RALF: A reinforced active learning formulation for object class recognition. In *CVPR*, pages 3626–3633. IEEE Computer Society, 2012.

[14] V. Franc, D. Průša, and V. Vorácek. Optimal strategies for reject option classifiers. *J. Mach. Learn. Res.*, 24:11:1–11:49, 2023.

[15] P. I. Frazier. A tutorial on bayesian optimization. *CoRR*, abs/1807.02811, 2018.

[16] K. Fukunaga and D. L. Kessell. Application of optimum error-reject functions (corresp.). *IEEE Trans. Inf. Theory*, 18(6):814–817, 1972.

[17] G. Hacohen, A. Dekel, and D. Weinshall. Active learning on a budget: Opposite strategies suit high and low budgets. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 8175–8195. PMLR, 2022.

[18] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao. Adbench: Anomaly detection benchmark. In *NeurIPS*, 2022.

[19] K. Hendrickx, L. Perini, D. V. der Plas, W. Meert, and J. Davis. Machine learning with a reject option: a survey. *Mach. Learn.*, 113(5):3073–3110, 2024.

[20] E. Hüllermeier and W. Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Mach. Learn.*, 110(3):457–506, 2021.

[21] D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *ICML*, pages 148–156. Morgan Kaufmann, 1994.

[22] R. A. Maxion and K. M. C. Tan. Benchmarking anomaly-based detection systems. In *DSN*, pages 623–630. IEEE Computer Society, 2000.

[23] R. M. Monarch. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Simon and Schuster, 2021.

[24] L. Perini and J. Davis. Unsupervised anomaly detection with rejection. *CoRR*, abs/2305.13189, 2023.

[25] L. Perini, V. Vercruyssen, and J. Davis. Class prior estimation in active positive and unlabeled learning. In *IJCAI*, pages 2915–2921. ijcai.org, 2020.

[26] L. Perini, V. Vercruyssen, and J. Davis. Quantifying the confidence of anomaly detectors in their example-wise predictions. In *ECML/PKDD (3)*, volume 12459 of *Lecture Notes in Computer Science*, pages 227–243. Springer, 2020.

[27] L. Perini, V. Vercruyssen, and J. Davis. Transferring the contamination factor between anomaly detection domains by shape similarity. In *AAAI*, pages 4128–4136. AAAI Press, 2022.

[28] L. Perini, P. Bürkner, and A. Klami. Estimating the contamination factor's distribution in unsupervised anomaly detection. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 27668–27679. PMLR, 2023.

[29] L. Perini, M. Rudolph, S. Schmedding, and C. Qiu. Uncertainty-aware evaluation of auxiliary anomalies with the expected anomaly posterior. *arXiv preprint arXiv:2405.13699*, 2024.

[30] T. Pimentel, M. Monteiro, A. Veloso, and N. Ziviani. Deep active learning for anomaly detection. In *IJCNN*, pages 1–8. IEEE, 2020.

[31] A. Pugnana and S. Ruggieri. Auc-based selective classification. In *AISTATS*, volume 206 of *Proceedings of Machine Learning Research*, pages 2494–2514. PMLR, 2023.

[32] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K. Müller, and M. Kloft. Deep semi-supervised anomaly detection. In *ICLR*. OpenReview.net, 2020.

[33] D. Shen, J. Zhang, J. Su, G. Zhou, and C. L. Tan. Multi-criteria-based active learning for named entity recognition. In *ACL*, pages 589–596. ACL, 2004.

[34] A. Sotgiu, A. Demontis, M. Melis, B. Biggio, G. Fumera, X. Feng, and F. Roli. Deep neural rejection against adversarial examples. *EURASIP J. Inf. Secur.*, 2020:5, 2020.

[35] C. D. Stefano, C. Sansone, and M. Vento. To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Trans. Syst. Man Cybern. Part C*, 30(1):84–94, 2000.

[36] L. Stradiotti, L. Perini, and J. Davis. Semi-supervised isolation forest for anomaly detection. In *SDM*, pages 670–678. SIAM, 2024.

[37] F. Tortorella. An optimal reject rule for binary classifiers. In *SSPR/SPR*, volume 1876 of *Lecture Notes in Computer Science*, pages 611–620. Springer, 2000.

[38] V. Vercruyssen, W. Meert, G. Verbruggen, K. Maes, R. Baumer, and J. Davis. Semi-supervised anomaly detection with an application to water analytics. In *ICDM*, pages 527–536. IEEE Computer Society, 2018.

[39] A. Zaher, S. McArthur, D. Infield, and Y. Patel. Online wind turbine fault detection through automated scada data analysis. *Wind Energy*, 12:574 – 593, 09 2009. doi: 10.1002/we.319.

[40] X. Zhan, H. Liu, Q. Li, and A. B. Chan. A comparative survey: Benchmarking for pool-based active learning. In *IJCAI*, pages 4679–4686. ijcai.org, 2021.

[41] L. Zhang, K. Liu, Y. Wang, and Z. B. Omariba. Ice detection model of wind turbine blades based on random forest classifier. *Energies*, 11(10): 2548, 2018.

[42] L. Zhou, F. Martínez-Plumed, J. Hernández-Orallo, C. Ferri, and W. Schellaert. Reject before you run: Small assessors anticipate big language models. In *EBeM@IJCAI*, volume 3169 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2022.

[43] K. Zope, K. Singh, S. Nistala, A. Basak, P. Rathore, and V. Runkana. Anomaly detection and diagnosis in manufacturing systems: A comparative study of statistical, machine learning and deep learning techniques. In *Annu. Conf. PHM Soc*, volume 11, 2019.