Deep Reinforcement Learning for Unknown Anomaly Detection

4 authors:

Guansong Pang
Singapore Management University
88 PUBLICATIONS 2,592 CITATIONS
SEE PROFILE

Anton van den Hengel
University of Adelaide
395 PUBLICATIONS 23,859 CITATIONS
SEE PROFILE

Chunhua Shen
University of Adelaide
503 PUBLICATIONS 44,521 CITATIONS
SEE PROFILE

Longbing Cao
University of Technology Sydney
379 PUBLICATIONS 9,180 CITATIONS
SEE PROFILE

Deep Reinforcement Learning for Unknown Anomaly Detection

Guansong Pang
Australian Institute for Machine Learning
University of Adelaide
Adelaide SA 5005, Australia
guansong.pang@adelaide.edu.au

Anton van den Hengel
Australian Institute for Machine Learning
University of Adelaide
Adelaide SA 5005, Australia
anton.vandenhengel@adelaide.edu.au

Chunhua Shen
Australian Institute for Machine Learning
University of Adelaide
Adelaide SA 5005, Australia
chunhua.shen@adelaide.edu.au

Longbing Cao
Advanced Analytics Institute
University of Technology Sydney
Ultimo NSW 2007, Australia
longbing.cao@uts.edu.au

Abstract

We address a critical yet largely unsolved anomaly detection problem, in which we aim to learn detection models from a small set of partially labeled anomalies and a large-scale unlabeled dataset. This is a common scenario in many important applications. Existing related methods either proceed unsupervised with the unlabeled data, or exclusively fit the limited anomaly examples that often do not span the entire set of anomalies. We propose here instead a deep reinforcement-learning-based approach that actively seeks novel classes of anomaly that lie beyond the scope of the labeled training data. This approach learns to balance exploiting its existing data model against exploring for new classes of anomaly. It is thus able to exploit the labeled anomaly data to improve detection accuracy, without limiting the set of anomalies sought to those given anomaly examples. This is of significant practical benefit, as anomalies are inevitably unpredictable in form and often expensive to miss. Extensive experiments on 48 real-world datasets show that our approach significantly outperforms five state-of-the-art competing methods.

1 Introduction

Anomaly detection finds application in a broad range of critical domains, such as intrusion detection in cybersecurity, early detection of disease in healthcare, and fraud detection in finance. Anomalies often exhibit diverse causes, which results in different types/classes of anomalies having distinctly heterogeneous features. For example, different types of network attack can embody entirely dissimilar underlying behaviors. By definition, anomalies also occur rarely, and unpredictably, in a dataset. It is therefore difficult, if not impossible, to obtain training data that covers all possible classes of anomaly. This renders supervised methods impractical. Unsupervised approaches have dominated in this area for decades for this reason [1, 2]. In many important applications, however, there exist a small set of known instances of important classes of anomalies. These labeled anomalies provide valuable prior knowledge, enabling significant accuracy improvements over unsupervised methods [3–7]. The challenge then is how to exploit those limited labeled anomaly examples without assuming that they illustrate every class of anomaly.

In most anomaly detection scenarios the volume of unlabeled data available is far more than could practically be processed. This unlabeled dataset is often arbitrarily truncated as a result. The approach

Preprint. Under review.

proposed here, however, is that we should actively select the unlabeled data that best informs our model. This inevitably involves a compromise between exploring the data for new classes on anomaly, or exploiting the existing data model to better detect the anomaly classes already identified.

In this work we consider the problem of anomaly detection with partially labeled anomaly data, *i.e.*, large-scale unlabeled data and a small set of labeled anomalies that only partially cover the classes of anomaly. Unsupervised anomaly detection approaches [4, 8–11] can often detect diverse anomalies because they are not limited by any labeled data, but they can produce many false positives due to the lack of prior knowledge of true anomalies. A few recently emerged semi-supervised approaches [6, 7, 12] aim to utilize those labeled data, but their models are exclusively fitted to the limited labeled anomalies, ignoring the supervisory signals from the possible anomalies in the unlabeled data. A straightforward solution to this issue is to use current unsupervised methods to detect some pseudo anomalies from the unlabeled data [4, 5], and then feed these pseudo anomalies and the labeled anomalies to learn more generalized abnormality using advanced detection models, *e.g.*, those in [6, 7, 12]. However, the pseudo labeling can have many false positives due to the limitation of unsupervised methods, leading to ineffective exploration and deteriorated exploitation; moreover, the labeling and the detection modeling are two decoupled steps, failing to jointly optimize the two steps.

To address the problem, this paper proposes an anomaly detection-oriented deep reinforcement learning (DRL) approach that actively seeks and learns novel classes of anomaly that lie beyond the scope of the labeled anomaly data. Particularly, a neural network-enabled *anomaly detection agent* is devised to exploit the labeled anomaly data to improve detection accuracy, without limiting the set of anomalies sought to those given anomaly examples. The agent achieves this by interacting with an environment created from the training data. Most real-world anomaly detection applications involve no sequential decision process (*e.g.*, tabular data), and thus, cannot provide the interactive environment. An *anomaly-biased simulation environment* is created to enable the agent to effectively exploit the small set of labeled anomalies while being deliberately explore the large-scale unlabeled data for any possible anomalies that lie outside the scope of this set. We further define a *reward* function using a synthesis of supervisory signals from both the labeled and suspicious unlabeled anomalies to achieve a balanced exploration-exploitation.

Contributions. In summary, this work makes the following two major contributions. (i) We introduce a novel DRL approach specifically designed for anomaly detection with partially labeled anomaly data. The resulting DRL agent is able to actively explore rare and novel unlabeled anomalies to learn abnormality beyond the scope of the given anomaly examples; its anomaly detection-oriented exploration and exploitation are jointly optimized through the agent-environment interactions. (ii) We instantiate the proposed framework into a detection model called DPLAN. The model is extensively evaluated on 48 datasets generated from four real-world datasets to replicate scenarios with known anomaly classes of different coverage. The results show that our model performs significantly better and more stably than five state-of-the-art semi-supervised and unsupervised methods, achieving respective 5%-8% and 34% *absolute* improvement in precision-recall rates.

2 Related Work

Anomaly Detection. Most conventional approaches [1, 8–10] are unsupervised without using any labeled data, but they are often ineffective when handling high-dimensional and/or intricate data. Recently deep learning has been explored to enhance the unsupervised detection, *e.g.*, by using data reconstruction [13–15], or learning feature representations tailored for specific anomaly measures [4, 11, 16]. Some early exploration [4, 6, 7, 12] show that deep anomaly detection can be substantially improved when some labeled anomalies are leveraged to guarantee a margin between the labeled anomalies and normal instances. Prior to that, a few studies utilize those anomaly examples, *e.g.*, by label propagation [3] or clustering [5]. One shared issue is that their models can be overwhelmingly dominated by the supervisory signals from the anomaly examples. There have been a number of deep methods based on, *e.g.*, adversarial training [17, 18], geometric feature transformation [19, 20], one-class classification [21–24], or predictive modeling [25, 26], but they assume the availability of large-scale labeled normal instances and thus address a different setting from ours (see [27] for a comprehensive survey of deep anomaly detection methods).

Additionally, our problem appears to be similar to PU (positive-unlabeled) learning [28–31], but they are two fundamentally different problems, because the positive instances (*i.e.*, anomalies) in our problem lie in different manifolds or class structures, whereas PU learning assumes the positive instances share the same manifold or class structure. Also, the exploration of unlabeled anomalies is related to active anomaly detection [32–35]. However, active anomaly detection approaches ask human experts to label a set of queried instances, and focus on devising cost-effective querying methods to support the labeling; by contrast, our approach aims at *automatically* exploring the unlabeled data, which is significantly more practical and applicable in real-world applications.

DRL for Knowledge Discovery. DRL has demonstrated human-level capability in several tasks, such as Atari 2600 games [36], the game of Go [37] and StarCraft [38]. Motivated by those tremendous success, DRL-driven real-world knowledge discovery emerges as a popular research area. Some successful application examples are recommender systems [39, 40] and automatic machine learning [41, 42]. A related application to anomaly detection is recently investigated in [43], in which *inverse reinforcement learning* [44] is explored for sequential anomaly detection. Our work is very different from [43] in that (i) they focus on unsupervised settings vs. our semi-supervised settings; (ii) a sequential decision process is assumed in [43], largely limiting its applications, whereas our approach does not have such assumptions; and (iii) they aim at learning an implicit reward function whereas we aim at leveraging predefined reward functions to train anomaly detection agents.

3 The Proposed Approach

3.1 Problem Statement

Given a training dataset $\mathcal{D} = \{\mathcal{D}^a, \mathcal{D}^u\}$ (with $\mathcal{D}^a \cap \mathcal{D}^u = \emptyset$), where $\mathcal{D}^a$ consists of a few labeled anomalies from a set of anomaly classes $\{C_1, C_2, \cdots, C_k\}$ while $\mathcal{D}^u$ is an unlabeled data set composed by normal instances and some anomalies drawn from a larger set of anomaly classes $\{C_1, C_2, \cdots, C_k, C_{k+1}, C_{k+2}, \cdots, C_{k+m}\}$, including unknown anomaly classes $\{C_{k+1}, C_{k+2}, \cdots, C_{k+m}\}$, our goal is to learn an anomaly scoring function $\phi : \mathcal{D} \mapsto \mathbb{R}$ that assigns anomaly scores to data instances so that $\phi(\mathbf{s}^i) > \phi(\mathbf{s}^j)$, where $\mathbf{s}^i, \mathbf{s}^j \in \mathbb{R}^D$ are data instances from $\mathcal{D}$, $\mathbf{s}^i$ is an anomaly and $\mathbf{s}^j$ is a normal instance.

3.2 Deep Reinforcement Learning Tailored for Anomaly Detection

Overview of Our Approach. We introduce an anomaly detection-oriented deep reinforcement learning approach that actively explores the unlabeled data to learn abnormality beyond the scope of the labeled anomalies. An overview of our framework is illustrated in Figure 1.

The framework aims to learn a neural network-enabled anomaly detection agent A that selects an optimal action out of two possible actions: a^0 and a^1, respectively corresponding to labeling a given *observation*[1] $\mathbf{s} \in \mathcal{D}$ as '*normal*' and '*anomalous*'. An anomaly-biased environment E is defined by a composition of an external handcrafted reward function h and an observation generator g to train the agent. Specifically, at each time step t, the agent A receives an observation $\mathbf{s}_t$ generated by the observation generator g and takes action a_t, and then receives an external reward r^e yielded by the reward function h, which is handcrafted to enforce the agent to correctly detect all labeled anomalies in $\mathcal{D}^a$ to maximize its cumulative future reward. Additionally, an intrinsic reward function f is defined to provide a reward r^i based on the novelty/abnormality of the observation $\mathbf{s}_t$, which is devised to encourage unsupervised active exploration of $\mathcal{D}^u$ for detecting possible unlabeled anomalies. The agent is iteratively trained in this manner with a number of episodes, with each *episode* consisting of a fixed number of observations.

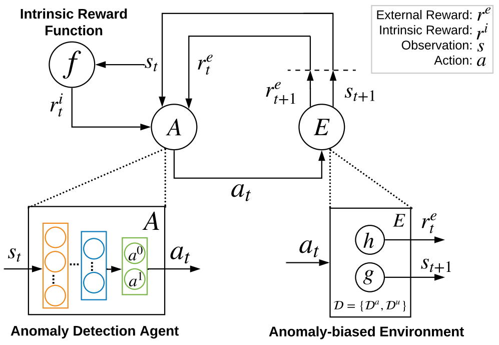

Figure 1: The Proposed Anomaly Detection-oriented Deep Reinforcement Learning Framework.

[1] The terms 'observation' and 'instance' are used interchangeably throughout the paper.

During training, the rewards the agent receives are positively associated with the detection of anomalies. Thus, at the inference stage, given a test observation $\hat{\mathbf{s}}$, its abnormality can be directly inferred based on the agent's estimated *value* (*i.e.*, the future rewards that can be expected) in taking action a^1 when observing $\hat{\mathbf{s}}$.

The proposed framework is instantiated into an anomaly detection model called Deep Q-learning with Partially Labeled ANomalies (**DPLAN**), which is introduced in detail as follows.

Anomaly Detection-oriented Agent. Our agent A aims to learn an optimal anomaly detection-oriented action-value function (*i.e.*, Q-value function), which can be approximated as:

$$Q^*(\mathbf{s}, a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | \mathbf{s}_t = \mathbf{s}, a_t = a, \pi], \tag{1}$$

which is the maximum expected return starting from an observation $\mathbf{s}$, taking the action $a \in \{a^0, a^1\}$, and thereafter following a behavior policy $\pi = P(a|\mathbf{s})$, with the *return* defined as the sum of rewards r_t discounted by a factor γ at each time step t. Different off-the-shelf DRL algorithms can be used to learn $Q^*(\mathbf{s}, a)$. In this work, the well-known deep Q-network (DQN) [36] is used, which leverages deep neural networks as the function approximator with the parameters θ: $Q(\mathbf{s}, a; \theta) = Q^*(\mathbf{s}, a)$ and learns the parameters θ by iteratively minimizing the following loss:

$$L_j(\theta_j) = \mathbb{E}_{(\mathbf{s},a,r,\mathbf{s}') \sim U(\mathcal{E})} \left[\left(r + \gamma \max_{a'} Q(\mathbf{s}', a'; \theta_j^-) - Q(\mathbf{s}, a; \theta_j) \right) \right], \tag{2}$$

where $\mathcal{E}$ is a set of the agent's learning experience with each element stored as $e_t = (\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$; the loss is calculated using minibatch samples drawn uniformly at random from the stored experience; θ_j are the parameters of the Q-network at iteration j; the network with the parameters θ_j^- is treated as a target network to compute the target at iteration j, having θ_j^- updated with θ_j every K steps.

Anomaly-biased Simulation Environment. We create a simulation environment E so that the agent A can automatically interact with E to exploit $\mathcal{D}^a$ while being actively explore $\mathcal{D}^u$.

Proximity-dependent Next Observation Sampling Function g. The sampling function g, a key module in E, is composed by two functions, g_a and g_u, to empower a balanced exploitation and exploration of the full data $\mathcal{D}$. Particularly, g_a is a function that uniformly samples $\mathbf{s}_{t+1}$ from $\mathcal{D}^a$ at random, *i.e.*, $\mathbf{s}_{t+1} \sim U(\mathcal{D}^a)$, which offers the same chance for each labeled anomaly to be exploited by the agent. g_u is a function that samples $\mathbf{s}_{t+1}$ from $\mathcal{D}^u$ based on the proximity of the current observation. To enable effective and efficient exploration of $\mathcal{D}^u$, g_u is defined as

$$g_u(\mathbf{s}_{t+1} | \mathbf{s}_t, a_t; \theta^e) = \begin{cases} \arg\min_{\mathbf{s} \in \mathcal{S}} d\big(\mathbf{s}_t, \mathbf{s}; \theta^e\big) & \text{if } a_t = a^1 \\ \arg\max_{\mathbf{s} \in \mathcal{S}} d\big(\mathbf{s}_t, \mathbf{s}; \theta^e\big) & \text{if } a_t = a^0, \end{cases} \tag{3}$$

where $\mathcal{S} \subset \mathcal{D}^u$ is a random subsample, θ^e are the parameters of $\psi(\cdot; \theta^e)$ which is a feature embedding function derived from the last hidden layer of our DQN, and d returns a Euclidean distance between $\psi(\mathbf{s}_t; \theta^e)$ and $\psi(\mathbf{s}; \theta^e)$ to capture the distance perceived by the agent in its embedding space.

Both of g_a and g_u are used in our simulator: with probability p the simulator performs g_a, and with probability $1 - p$ the simulator performs g_u. This way enables the agent to sufficiently exploit the small labeled anomaly data while exploring the large unlabeled data. In this work $p = 0.5$ is used to balance the exploration-exploitation.

In Eq. (3), g_u returns the nearest neighbor of $\mathbf{s}_t$ when the agent believes the current observation $\mathbf{s}_t$ is an anomaly and takes action a^1. This way allows the agent to explore observations that are similar to the suspicious anomaly observations. g_u returns the farthest neighbor of $\mathbf{s}_t$ when A believes $\mathbf{s}_t$ is a normal observation and takes action a^0, in which case the agent explores potential anomaly observations that are far away from the normal observation. Thus, both cases are served for effective active exploration of the possible anomalies in the large $\mathcal{D}^u$.

The parameters θ^e are a subset of the parameters θ in DQN. The nearest and farthest neighbors are approximated on subsample $\mathcal{S}$ rather than $\mathcal{D}^u$ for efficiency consideration, and we found empirically that the approximation is as effective as performing g_u on the full $\mathcal{D}^u$. $|\mathcal{S}| = 1000$ is set by default. $\mathcal{S}$ and θ^e are constantly updated to compute d for each step.

Anomaly-biased External Handcrafted Reward Function h. The below h function is defined in the environment simulator to yield a handcrafted reward signal r_t^e to our agent:

$$r_t^e = h(\mathbf{s}_t, a_t) = \begin{cases} 1 & \text{if } a_t = a^1 \text{ and } \mathbf{s}_t \in \mathcal{D}^a \\ 0 & \text{if } a_t = a^0 \text{ and } \mathbf{s}_t \in \mathcal{D}^u \\ -1 & \text{otherwise.} \end{cases} \tag{4}$$

It indicates that the agent receives a positive r^e only when it correctly labels the known anomalies as '*anomalous*'. Thus, r^e explicitly encourages the agent to fully exploit the labeled data $\mathcal{D}^a$.

Overall Reward. In addition to the external reward r^e yielded in Eq. (4), we introduce an intrinsic reward r^i measured by the novelty/abnormality of an observation in an unsupervised way, a.k.a. the agent's curiosity [45, 46]. Unlike r^e that encourages the exploitation of the labeled data $\mathcal{D}^a$, r^i is devised to encourage the agent to explore anomalies in the unlabeled data $\mathcal{D}^u$ and defined as

$$r_t^i = f(\mathbf{s}_t; \theta^e) = \text{iForest}\big(\mathbf{s}_t; \theta^e\big), \tag{5}$$

where f measures the abnormality of $\mathbf{s}_t$ using the well-known isolation-based unsupervised anomaly detector, iForest [8]. Isolation is defined by the number of steps required to isolate an observation $\mathbf{s}$ from the observations in $\mathcal{D}^u$ through half-space data partition. iForest is used here because it is computationally efficient and excels at identifying rare and heterogeneous anomalies.

Similar to g_u in Eq. (3), the f function also operates on the low-dimensional ψ embedding space parameterized by θ^e. That means both the training and inference in iForest are performed using the ψ-based projected data. This enables us to capture the abnormality that is faithful w.r.t. our agent. This also guarantees iForest always works on low-dimensional space as it fails to work effectively in high-dimensional space [8]. The output of iForest is rescaled into the range $[0, 1]$, we accordingly have $r^i \in [0, 1]$, with larger r^i indicating more abnormal. Thus, regardless of the action taken, our agent would receive large r^i whenever the agent believes the observation is rare or novel compared to previously seen observations. This way helps the agent detect possible unlabeled anomalies in $\mathcal{D}^u$.

To balance the importance of exploration and exploitation, the overall reward the agent receives at each time step t is defined as

$$r_t = r_t^e + r_t^i. \tag{6}$$

Anomaly Detection Using DPLAN. During training, the agent A in DPLAN is trained to minimize the loss in Eq. (2) in an end-to-end fashion. Let $Q(\mathbf{s}, a; \theta^*)$ be the Q-network with the learned θ^* after training, then at the inference stage, $Q(\hat{\mathbf{s}}, a; \theta^*)$ outputs an estimated value of taking action a^0 or a^1 given a test observation $\hat{\mathbf{s}}$. Since a^1 corresponds to the action of labeling $\hat{\mathbf{s}}$ as '*anomalous*', $Q(\hat{\mathbf{s}}, a^1; \theta^*)$ can be used as anomaly score. The intuition behind this scoring is discussed as follows.

Let π be a policy derived from Q, then the expected return of taking the action a^1 given the observation $\hat{\mathbf{s}}$ under the policy π, denoted by $q_\pi(\hat{\mathbf{s}}, a^1)$, can be defined as

$$q_\pi(\hat{\mathbf{s}}, a^1) = \mathbb{E}_\pi\left[\sum_{n=0}^{\infty} \gamma^n r_{t+n+1} \middle| \hat{\mathbf{s}}, a^1\right]. \tag{7}$$

Let $\hat{\mathbf{s}}^i$, $\hat{\mathbf{s}}^j$ and $\hat{\mathbf{s}}^k$ be labeled anomalies, unlabeled anomalies and unlabeled normal observations respectively, we would have $h(\hat{\mathbf{s}}^i, a^1) > h(\hat{\mathbf{s}}^j, a^1) > h(\hat{\mathbf{s}}^k, a^1)$ and $f(\hat{\mathbf{s}}^i; \theta^e) \approx f(\hat{\mathbf{s}}^j; \theta^e) > f(\hat{\mathbf{s}}^i; \theta^e)$. Since r_t in Eq. (7) is the sum of the outputs of the h and f functions, $q_\pi(\hat{\mathbf{s}}^i, a^1) > q_\pi(\hat{\mathbf{s}}^j, a^1) > q_\pi(\hat{\mathbf{s}}^k, a^1)$ holds under the same policy π. Thus, when the agent well approximates the Q-value function, its estimated returns would be: $Q(\hat{\mathbf{s}}^i, a^1; \theta^*) > Q(\hat{\mathbf{s}}^j, a^1; \theta^*) > Q(\hat{\mathbf{s}}^k, a^1; \theta^*)$; so the observations with large $Q(\hat{\mathbf{s}}, a^1; \theta^*)$ are anomalies of our interest. See Appendix A in the **Supplementary Material** for detailed algorithmic description of DPLAN.

4 Experiments

4.1 Datasets

Four widely-used real-world datasets that contain two-to-seven classes of (semantically) real anomalies are used in our experiments. These include: *NB15* [6, 47] that contains 107,687 data instances in a 196-dimensional space from seven abnormal intrusion types and normal network flows, *Thyroid*

[6–8, 48] that contains 7,049 instances in a 21-dimensional space from hypothyroid/subnormal and normal thyroid patients, *HAR* [49, 50] that contains 7,707 instances in a 561-dimensional space from anomalous walking-downstairs/upstairs sensor data against the other usual human activity data (*e.g.*, sitting, standing, laying), and *Covertype* [8, 48, 50–52] that contains 296,633 instances in a 54-dimensional space from anomalous cottonwood and douglas-fir forest cover types against the common lodgepole pine. These four datasets serve as a base pool to create 48 datasets to evaluate the performance of DPLAN in scenarios with known anomaly classes of different coverage.

Scenario I: One Known Anomaly Class. We split each of the *NB15*, *Thyroid*, *HAR* and *Covertype* datasets into training and test sets, with 80% data of each class into the training data and the other 20% data into the test set. For the training data we retain only a few labeled anomalies to be $\mathcal{D}^a$, and randomly sample anomalies from each anomaly class and mix them with the normal training instances to produce the large anomaly-contaminated unlabeled data $\mathcal{D}^u$. We then create 13 datasets, with each dataset having $\mathcal{D}^a$ sampled from only *one specific anomaly class*; these datasets are shown in Table 1, where each dataset is named by the known anomaly class. The test data is fixed after the training-test data split, which contains one known and one-to-six unknown anomaly classes, accounting for 0.96%-5.23% of the test data. See Appendix B for more details of the 13 datasets.

Since only a small number of labeled anomalies are available in many applications, in each dataset the number of labeled anomalies is fixed to 60, accounting for 0.03%-1.07% training data only. Anomalies are rare events, so the anomaly contamination rate in $\mathcal{D}^u$ is fixed to 2%. Similar results can be observed with other contamination rates (see Appendix D for detail).

Scenario II: Increasing Number of Known Anomaly Classes. We also examine the scenarios with more known anomaly classes. This experiment focuses on the seven *NB15* datasets in Table 1, since it is inapplicable to *Thyroid*, *HAR* and *Covertype* that contain two anomaly classes only. Particularly, each of these seven datasets is used as a base, and a new randomly selected anomaly class with 60 anomalies is incrementally added into its $\mathcal{D}^a$ each step. This results in additional 35 datasets where each training data contains two-to-six known anomaly classes. The test data remains unchanged.

4.2 Competing Methods and Performance Evaluation

DPLAN is compared with five state-of-the-art semi/un-supervised anomaly detectors below:

- **DevNet** [6] is a deep semi-supervised method that leverages a few labeled anomalies and a Gaussian prior over anomaly scores to perform end-to-end anomaly detection.
- **Deep SAD** [7] is a deep semi-supervised method built upon the availability of a small number of labeled normal and anomalous instances. Following [7, 53], Deep SAD is adapted to our setting by enforcing a margin between the one-class center and the labeled anomalies while minimizing the center-oriented hypersphere.
- **REPEN** [4] is a recent deep unsupervised detector that learns representations specifically tailored for distance-based anomaly measures. Another popular deep unsupervised detector DAGMM [11] is also tested, but it is less effective than REPEN. Thus we focus on REPEN.
- **iForest** [8] is a widely-used unsupervised method that detects anomalies based on how many steps are required to isolate the instances by random half-space partition in isolation trees.
- **DevNet$^+$** is DevNet trained with the labeled anomaly set and some pseudo anomalies identified in the unlabeled data. To have a straightforward comparison, DevNet$^+$ uses the same unlabeled anomaly explorer as DPLAN: iForest. iForest returns a ranking of data instances only. A cutoff threshold, *e.g.*, the top-ranked $n\%$ instances, is required to obtain the pseudo anomalies. $\{0.01\%, 0.05\%, 0.1\%, 0.5\%, 1\%, 2\%, 4\%\}$ are probed. We report the best performance achieved using the threshold 0.05%.

A multilayer perceptron network is used in the Q-network since the experiments focus on tabular data. All competing deep methods worked effectively using one hidden layer but failed to work using a deeper network due to the limit of the small labeled data. To have a pair comparison, all deep methods use one hidden layer with l units and the ReLU activation [54] by default. Following [4, 6], $l = 20$ is used. DPLAN performs similarly well with other l settings (see Appendix E for detail). It can also work effectively with deeper architectures (see ablation study in Section 4.4 for detail).

DPLAN is trained with 10 episodes by default, with each episode consisting of 2,000 steps. 10,000 warm-up steps are used. The target network in DQN is updated every $K = 10,000$ steps. The other optimization settings of DPLAN are set to the default settings in the original DQN. DevNet and REPEN are used with the settings respectively recommended in [4, 6]. Deep SAD uses the same optimization settings as DevNet, which enable it to obtain the best performance. The isolation trees with the same settings recommended in [8] are used in iForest, Eq. (5) in DPLAN and DevNet$^+$. See Appendix C for more details of implementing DPLAN and its five competing methods.

Two widely-used complementary performance metrics, Area Under Receiver Operating Characteristic Curve (AUC-ROC) and Area Under Precision-Recall Curve (AUC-PR) [55], are used. All reported results are averaged over 10 independent runs. The paired *Wilcoxon* signed rank [56] using AUC-ROC (AUC-PR) across multiple datasets is used to examine the statistical significance of the results.

4.3 Comparison to State-of-the-art Methods

Scenario I. The comparison results on the 13 datasets with one known anomaly class are shown in Table 1. DPLAN achieves the best performance on 10 datasets in both AUC-PR and AUC-ROC. Particularly, in AUC-PR, on average, DPLAN outperforms the deep semi-supervised detectors DevNet, Deep SAD and DevNet$^+$ by 5%-8%, and both of the unsupervised anomaly detectors by 34%. DPLAN is also the best performer in AUC-ROC, outperforming all competing methods by 1%-10%. The improvement of DPLAN in AUC-PR over all counterparts is significant at the 99% confidence level; the improvement in AUC-ROC is also significant at least at the 90% confidence level. Furthermore, DPLAN performs very stably across all 13 datasets and has substantially smaller AUC standard deviation than all five competing methods.

Table 1: AUC-PR and AUC-ROC Results (mean±std %) on 13 Real-world Datasets.

Data		AUC-PR Performance						AUC-ROC Performance					
Base	**Dataset**	**DPLAN**	**DevNet**	**DeepSAD**	**REPEN**	**iForest**	**DevNet$^+$**	**DPLAN**	**DevNet**	**DeepSAD**	**REPEN**	**iForest**	**DevNet$^+$**
NB15	Analysis	**68.3**±0.8	64.0±3.3	59.5±3.6	44.7±0.8	37.8±2.2	60.9±1.0	**85.2**±0.4	83.9±5.2	76.9±1.9	81.0±1.9	73.8±1.8	84.7±1.0
	Backdoor	70.0±0.4	70.2±3.4	67.8±5.7	38.9±0.7	37.1±2.5	**71.3**±0.4	**83.5**±0.6	79.5±6.1	75.3±4.9	80.4±1.5	73.6±2.1	80.7±0.9
	DoS	68.1±0.2	71.8±2.2	69.0±2.8	36.5±1.1	37.9±2.7	**75.1**±1.1	80.9±0.5	84.6±2.4	77.9±3.0	75.7±1.8	73.7±2.4	**87.1**±1.1
	Exploits	**76.8**±0.4	66.0±4.6	63.6±3.9	37.3±0.7	36.7±2.6	58.9±7.0	**90.6**±0.4	87.1±2.8	79.8±2.9	77.4±2.1	73.2±2.2	85.8±1.2
	Fuzzers	**64.6**±0.8	49.3±2.8	49.9±5.8	36.1±0.6	37.1±2.5	50.0±5.4	**87.8**±0.2	84.1±0.4	83.9±1.0	76.7±1.5	73.5±1.9	85.6±1.5
	Generic	**75.9**±0.4	74.8±3.9	70.3±1.3	48.5±0.7	38.0±2.6	64.7±2.1	82.7±0.7	82.0±3.2	79.3±2.6	**85.4**±0.7	73.7±2.0	82.8±1.4
	Recon	43.8±0.9	38.6±0.4	39.2±0.5	**45.7**±0.8	37.4±2.4	41.4±1.4	80.9±0.8	81.9±0.2	82.1±0.3	80.9±1.4	73.5±1.2	**82.9**±0.5
Thyroid	Hypothyroid	**49.0**±0.1	46.9±0.5	39.8±1.2	8.1±0.3	15.5±2.0	43.2±1.1	**84.6**±0.1	83.5±0.3	80.9±0.5	53.6±0.9	68.3±2.3	82.8±0.5
	Subnormal	**43.6**±0.7	37.9±3.1	30.8±3.5	7.9±0.2	18.4±2.8	28.8±3.2	**82.1**±0.1	78.4±0.8	75.8±0.5	52.3±0.9	73.3±2.3	80.0±0.9
HAR	Downstairs	**94.3**±0.1	87.4±2.5	88.7±1.8	30.0±0.5	36.8±1.6	84.4±2.2	**99.3**±0.1	99.0±0.4	99.1±0.4	91.1±0.6	92.6±0.5	99.1±0.2
	Upstairs	**94.2**±0.5	86.5±0.9	88.7±0.8	29.7±0.4	39.4±1.9	90.0±1.1	**99.6**±0.1	98.3±0.9	99.0±0.1	91.8±0.6	94.0±0.4	99.4±0.1
Covertype	Cottonwood	**70.9**±0.1	67.0±2.2	67.8±2.8	42.4±4.1	44.3±6.9	59.3±2.3	**92.3**±0.2	86.8±1.9	87.6±4.2	89.1±1.9	84.9±2.7	84.1±3.2
	Douglas-fir	**77.6**±0.3	72.2±1.9	72.8±1.4	45.3±2.1	45.6±7.5	66.9±1.1	**97.6**±0.0	97.4±0.3	97.3±0.2	90.1±1.0	86.2±2.8	96.3±0.2
	Average	**69.0**±0.4	64.1±2.4	62.1±2.7	34.7±1.0	35.5±3.1	61.3±2.3	**88.2**±0.3	86.7±1.9	84.2±1.7	78.9±1.3	78.0±1.9	87.1±1.0
	P-value	-	0.0024	0.0005	0.0005	0.0002	0.0034	-	0.0254	0.0017	0.0010	0.0002	0.0769

Scenario II. The comparison results with increasing number of known anomaly classes are given in Figure 2. REPEN and iForest are insensitive to the increased anomalies, so their results in Table 1 are used as baselines. In general, increasing the coverage of known anomalies provide more supervision information, which enables DPLAN, DevNet, Deep SAD and DevNet$^+$ to achieve considerable improvement, especially on datasets where the first known anomaly class cannot provide much generalizable information, *e.g.*, *Fuzzers* and *Recon*. The AUC-PR of DPLAN increases remarkably from 43.8%-76.8% up to 82.6%-85.3% across the datasets, with maximal relative improvement as large as more than 91%. Although DPLAN is less effective than, or entangled with, DevNet, Deep SAD and DevNet$^+$ at the starting point on some datasets such as *Backdoor*, *DoS* and *Generic*, it evolves quickly and outperforms them by about 4%-6% in the end.

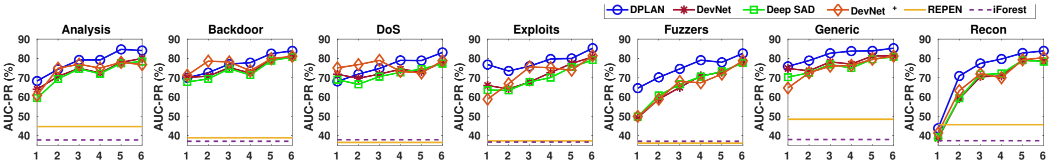

Figure 2: AUC-PR Results w.r.t. the Number of Known Anomaly Classes.

DRL Exploration vs. Pseudo Anomaly Labeling. As shown in Table 1 and/or Figure 2, the pseudo anomaly labeling in DevNet$^+$ helps improve DevNet in a few datasets such as *Backdoor*, *DoS*, *Recon* and *Upstairs*. However, it significantly degrades DevNet in many other datasets, especially in AUC-PR, because the pseudo anomalies can contain many false positives, which deteriorate the

DevNet's exploitation of the labeled anomalies. Although DPLAN and DevNet$^+$ use exactly the same unsupervised detector to explore the unlabeled data, the DRL exploration enables DPLAN to have active data exploration and simultaneously balance the exploitation and exploration, achieving substantially more effective exploration (see Section 4.4 for detail).

4.4 Analysis of DPLAN

Ablation Study. Below we analyze the contribution of three key components of DPLAN.

Anomaly-biased Simulation Environment. We compare DPLAN with its simplified variant, named **REnv**, in which the anomaly-biased observation generator g is replaced with a Random Environment (REnv), *i.e.*, next observations are randomly sampled from $\mathcal{D}$. As shown in Table 2, DPLAN outperforms REnv by more than 22% and 13% in average AUC-PR and AUC-ROC respectively, demonstrating the significant contribution of the anomaly-biased environment defined in DPLAN.

DRL Exploration. We also compare DPLAN with its variant, **ERew**, which is DPLAN with the External Reward (ERew) only, *i.e.*, no intrinsic reward. As shown in Table 2, DPLAN performs much better than ERew in most cases, significantly outperforming ERew in AUC-ROC at the 90% confidence level. This indicates that the isolation-based intrinsic reward-driven DRL exploration enables DPLAN to explore the unlabeled data effectively, providing important complementary supervision signals to the DPLAN's exploitation of the labeled data. Nevertheless, the DRL Exploration inversely affects the performance on two datasets *Generic* and *Cottonwood*, demonstrating the significant challenges underlying the exploration of the rare and heterogeneous unlabeled anomalies.

Network Architecture. **DQN$^+$** is DPLAN using a deeper Q-network, with two additional hidden layers with respective 500 and 100 ReLU units are added. Impressively, as shown in Table 2, DQN$^+$ achieves remarkably improvement over DPLAN. This is encouraging because it indicates DPLAN can learn more complex yet well generalized models from the limited labeled data, especially when the amount of unlabeled data is large, *e.g.*, the seven *NB15* datasets and the two *Covertype* datasets, whereas prior methods like DevNet drop significantly when using a deeper architecture [6].

Learning with More Training Steps. We further investigate the performance of DPLAN w.r.t. the number of training steps. The results are given in Figure 3. It shows that the AUC-PR and episode reward of DPLAN often converge very early, *e.g.*, around 20,000 training steps. It is interesting that through the 100,000 training steps, DPLAN is continuously enhanced on the *Hypothyroid* and *Subnormal* datasets, achieving as large as further 23% and 98% AUC-PR improvement compared to the version trained with 20,000 steps. This indicates that with larger training steps, DPLAN achieves much better exploration on these two datasets. However, the opposite may occur on the three datasets *Fuzzers*, *Generic* and *Recon*. DPLAN trained with 20,000 steps is generally recommended.

Table 2: DPLAN (Org) and Its Three Variants

	AUC-PR Results (%)				AUC-ROC Results (%)			
Dataset	**Org**	**ERew**	**REnv**	**DQN$^+$**	**Org**	**ERew**	**REnv**	**DQN**
Analysis	68.3	66.1	67.4	**77.6**	85.2	85.4	79.5	**91.1**
Backdoor	70.0	68.0	67.7	**82.0**	83.5	76.9	77.6	**92.1**
DoS	68.1	67.6	69.6	**75.8**	80.9	77.9	80.6	**91.7**
Exploits	**76.8**	75.1	67.9	72.0	**90.6**	90.3	78.5	89.8
Fuzzers	64.6	63.7	**69.7**	68.6	87.8	86.7	80.8	**88.7**
Generic	75.9	**77.9**	64.3	72.9	82.7	**86.9**	73.7	83.2
Recon	43.8	44.6	**66.5**	54.0	80.9	78.7	76.6	**87.9**
Hypothyroid	**49.0**	48.5	11.9	44.3	**84.6**	84.4	63.3	81.6
Subnormal	43.6	**44.7**	11.8	24.5	**82.1**	81.8	62.4	75.8
Downstairs	**94.3**	94.1	26.8	86.9	**99.3**	98.1	80.0	98.9
Upstairs	**94.2**	92.7	19.3	88.1	**99.6**	98.3	69.5	99.1
Cottonwood	70.9	**72.2**	36.7	70.0	92.3	93.9	80.7	**93.7**
Douglas-fir	**77.6**	76.5	30.0	75.2	**97.6**	97.5	74.7	97.5
Average	**69.0**	68.6	46.9	68.6	88.2	87.4	75.2	**90.1**
P-value	-	0.449	0.022	1.000	-	0.090	0.001	0.281

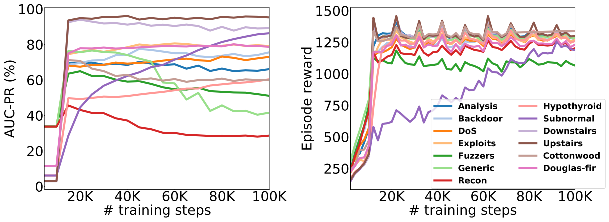

Figure 3: Results w.r.t. the Number of Training Steps

Computational Efficiency[2]. The time complexity of training DPLAN using stochastic optimization is constant w.r.t. data size and is linear to the number of training steps. It takes averagely 621 seconds to train DPLAN with 20,000 steps for all the 13 datasets in Table 1, which is slower than the competing methods which take 15-120 seconds. In practice the model training can be easily taken offline. The online inference runtime is normally much more important. Similar to DevNet, Deep SAD and DevNet$^+$, DPLAN takes a single forward-pass to obtain the anomaly scores, so they

[2] All runtimes are calculated under the environment: Intel Core i7-8700 CPU @ 3.20GHz x 12, 16GB RAM.

have similar inference complexity, *e.g.*, they all takes two to three seconds to complete the anomaly scoring of over 27,5000 test instances in total in all of the 13 datasets, which is much faster than REPEN and iForest that respectively takes about 40 and 20 seconds.

5 Conclusions

This paper proposes an anomaly detection-oriented deep reinforcement learning approach. Our approach can not only well exploit the limited labeled anomaly data, but also simultaneously actively explore the sparse and heterogeneous anomaly signals in the large unlabeled data, achieving significantly more generalized abnormality than existing methods on 48 real-world datasets. This also allows us to build more effective models with a deeper network architecture. Impressively, our approach can achieve further 23%-98% relative AUC-PR improvement by only increasing the number of training steps on some datasets. Its inference is also computationally efficient to scale. We are exploring methods to automate our model for any individual datasets to release its full potential.

Broader Impact

Anomaly detection could be applied to a broad range of real-world applications, including detection of network attacks/malware, credit card/insurance frauds, criminal activities, mechanical faults/defects, abnormal patient symptoms, and many more [1, 2]. Our research focuses particularly on a common scenario of these applications, in which partially labeled anomalies are available during training. Learning from those labeled anomalies generally helps substantially improve the detection recall rates and prevent many false-positive detection results, resulting in tremendous benefits, *e.g.*, in preventing crimes and undesired incidents in both the digital and physical worlds, better healthcare, etc. Additionally, anomaly detection (and hence our work) could also be applicable to a range of other relevant machine learning tasks, such as out-of-distribution example detection, adversarial example detection, curiosity learning and open-set recognition.

By definition, anomaly detection aims at identifying rare data instances, so a potential major risk of applying anomaly detection to real-world systems is the possible algorithmic bias against the minority groups presented in the data, such as the under-represented groups in fraud detection and crime detection systems [57, 58]. Our approach can mitigate some parts of this risk by informing the model to detect the anomalies of our interest, rather than simply rare instances, by training the model with some known anomaly examples. However, this risk may still exist in the exploration of the unlabeled data in our model. It could be further mitigated by using our model together with anomaly explanation algorithms [59, 60] that could provide practical explanation to why a specific instance is identified as anomaly.

A The Algorithmic Details of DPLAN

The full procedure of training DPLAN is presented in Algorithm 1. The first three steps initialize the size of the experience set and weight parameters of Q-value functions Q and $\hat{Q}$. DPLAN is then trained with $n\_episodes$ episodes, with each episode having n_steps training steps. For each episode, the first observation $\mathbf{s}_1 \sim U(\mathcal{D}^u)$ is uniformly sampled at random from the unlabeled data $\mathcal{D}^u$. Then in Steps 7-8, we adopt the same ϵ-greedy exploration as in the original DQN, in which with a probability of ϵ the agent randomly selects an action from $\{a^0, a^1\}$, and otherwise selects the action that maximizes the action-value function at the current time step. After the agent performing the selected action, the environment responses to the agent with next observation $\mathbf{s}_{t+1}$, with probability p we randomly sample it from the labeled anomaly set $\mathcal{D}^a$, *i.e.*, $\mathbf{s}_{t+1} \sim U(\mathcal{D}^a)$, and otherwise return the nearest/farthest neighbor of $\mathbf{s}_t$ in a random subsample $\mathcal{S} \subset \mathcal{D}^u$ based on $g_u(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t; \theta^e)$, where θ^e is a subset of parameters in θ and is constantly updated. The environment also gives a reward r^e_t to the agent. At the same time, $f(\mathbf{s}_t, \hat{\theta}^e)$ is used to yield an intrinsic reward r^i_t. $\hat{\theta}^e$ is exactly the same set of parameters as θ^e. $\hat{\theta}^e = \theta^e$ is updated every N steps rather than every step. Constantly updating $\hat{\theta}^e = \theta^e$ requires to frequently project data onto low-dimensional space and build iForest, which adds remarkably extra computation. The overall reward the agent receives in each time step is $r_t = r^e_t + r^i_t$. After that, we gain an experience record $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$ and store it into the experience set $\mathcal{E}$. Steps 14-16 then perform the Q-learning update, with the target action-value function $\hat{Q} = Q$ updated every K steps.

Algorithm 1 *Training DPLAN*

```
Input: $\mathcal{D} = \{\mathcal{D}^a, \mathcal{D}^u\}$ - training data
Output: $Q(\mathbf{s}, a; \theta^*)$ - action-value function (anomaly detection agent)
 1: Initialize action-value function $Q$ with random weights $\theta$
 2: Initialize target action-value function $\hat{Q}$ with weights $\theta^- = 0$
 3: Initialize the size of experience set $\mathcal{E}$ to $M$
 4: for $j = 1$ to $n\_episodes$ do
 5:     Initial observation $\mathbf{s}_1 \sim U(\mathcal{D}^u)$
 6:     for $t = 1$ to $n\_steps$ do
 7:         With probability $\epsilon$ select a random action $a_t$ from $\{a^0, a^1\}$
 8:         Otherwise select $a_t = \arg\max_a Q(\mathbf{s}_t, a; \theta)$
 9:         With probability $p$ the environment returns $\mathbf{s}_{t+1} \sim U(\mathcal{D}^a)$
10:         Otherwise return $\mathbf{s}_{t+1} \sim \mathcal{D}^u$ based on $g_u(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t; \theta^e)$
11:         Calculate intrinsic reward $r_t^i = f(\mathbf{s}_t, \hat{\theta}^e)$
12:         Receive reward $r_t = r_t^e + r_t^i$
13:         Store experience $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$ in $\mathcal{E}$
14:         Randomly sample a minibatch of experience records $(\mathbf{s}_l, a_l, r_l, \mathbf{s}_{l+1})$ from $\mathcal{E}$
15:         Set $y_l = \begin{cases} r_l & \text{if episode terminates at step } l+1 \\ r_l + \gamma \max_{a'} \hat{Q}(\mathbf{s}_{l+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
16:         Perform a gradient descent step on $\big(y_l - Q(\mathbf{s}_l, a_l; \theta)\big)^2$ w.r.t. the weight parameters $\theta$
17:         Update $\hat{\theta}^e = \theta^e$ every $N$ steps
18:         Update $\hat{Q} = Q$ every $K$ steps
19:     end for
20: end for
21: return $Q$
```

After training, DPLAN returns $Q(\mathbf{s}, a; \theta^*)$, which is an approximated optimal action-value function and can be seen as an anomaly detection agent to detect anomalies. The procedure of using DPLAN to detect anomalies in a test set $\mathcal{T}$ is presented in Algorithm 2. Specifically, given every observation $\mathbf{s}^j \in \mathcal{T}$, DPLAN performs one forward-pass in its network and then gets the estimated action-value for each action. If $\mathbf{s}^j$ is believed to be an anomaly, DPLAN would select action a^1 with a large action-value, and select a^0 with a small action-value otherwise. The observations with large action-value are considered as anomalies.

Algorithm 2 *Anomaly Detection using DPLAN*

```
Input: $\mathcal{T}$ - test data, $Q(\mathbf{s}, a; \theta^*)$ - anomaly detection agent
Output: $\mathbf{y}$ - anomaly scores
for $j = 1$ to $|\mathcal{T}|$ do
    $y^j = \arg\max_a Q(\mathbf{s}^j, a; \theta^*)$, $\mathbf{s}^j \in \mathcal{T}$
end for
return Anomaly scores $\mathbf{y}$
```

B Datasets

Four widely-used real-world datasets that contain two-to-seven classes of (semantically) real anomalies are used in our experiments, including *NB15*, *Thyroid*, *HAR* and *Covertype*. These four datasets are publicly available and can be accessed via the links given in Table 3.

NB15 is a recently released network intrusion datasets with a range of network attacks. The seven most common types of attacks, including *analysis*, *backdoor*, *DoS*, *exploits*, *fuzzers*, *generic* and *reconnaissance* (*recon* for short), are used as anomalies against normal network flows.

Thyroid is a dataset for detection of thyroid diseases, in which patients diagnosed with *hypothyroid* or *subnormal* are anomalies against normal patients.

Table 3: Links for accessing the datasets

Data	Link
NB15	https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/
Thyroid	https://www.openml.org/d/40497
HAR	https://www.openml.org/d/1478
Covertype	https://archive.ics.uci.edu/ml/datasets/covertype

HAR contains embedded inertial sensor data from a waist-mounted smartphone for six different human activities. The activities of walking downstairs and walking upstairs (*downstairs* and *upstairs* for short) are treated as abnormal activities w.r.t. the other four common activities.

Covertype contains cartographic data of seven forest cover types. Following the literature [8, 48, 50], the most dominant cover type *lodgepole pine* is used as the normal class against *cottonwood* and *douglas-fir* that demonstrates obvious deviations from lodgepole pine.

One-hot encoding is used to convert all categorical features into numeric features. Missing values are replaced with the mean value if there are any. All features are normalized into the range $[0, 1]$ before modeling. Anomalies, by definitions, are rare data instances. Thus, following the literature [6, 8, 48, 50, 61], random downsampling without replacement is applied to the *DoS*, *exploits*, *fuzzers*, *generic*, *recon*, *downstairs* and *upstairs* classes to guarantee the rarity nature of anomalies. Specifically, we downsample the *DoS*, *exploits*, *fuzzers*, *generic*, *recon* anomaly classes to have 3,000 instances so that all anomaly classes in the NB15 data are of a similar size. This is to guarantee the class balance among the anomaly classes to enable fair evaluation of the performance in detecting anomalies from different anomaly classes. The *downstairs* and *upstairs* are downsampled so that the anomalies from each of these classes account for 2% of the dataset. An overview of the key statistics of the four data bases is shown in Table 4.

Table 4: Key statistics of four data bases. D is the dimensionality. Each data base contains two-to-seven anomaly classes.

Data Base		**Normal Class**		**Anomaly Class**	
Data Name	D	Class Name	Class Size	Class Name	Class Size (%)
NB15	196	normal network flows	93,000	analysis	2,677 (2.80%)
				backdoor	2,329 (2.44%)
				DoS	3,000 (3.13%)
				exploits	3,000 (3.13%)
				fuzzers	3,000 (3.13%)
				generic	3,000 (3.13%)
				recon	3,000 (3.13%)
Thyroid	21	normal patients	6,666	hypothyroid	166 (2.43%)
				subnormal	368 (5.23%)
HAR	561	walking, sitting, standing, laying	7,349	downstairs	150 (2.00%)
				upstairs	150 (2.00%)
Covertype	54	the largest class (lodgepole pine)	283,301	cottonwood	2,747 (0.96%)
				douglas-fir	17,367 (5.78%)

To replicate the scenarios where we have some known anomaly examples in the training data and test data contains both known and unknown anomaly classes, we use the above four datasets as a base pool to further create 13 datasets, with each dataset containing one known anomaly class in its training data. Specifically, we split each of the *NB15*, *Thyroid*, *HAR* and *Covertype* datasets into training and test sets, with 80% data of each class into the training data and the other 20% data into the test set. For the training data we retain only a few labeled anomalies to be $\mathcal{D}^a$, and randomly sample anomalies from each anomaly class and mix them with the normal training instances to produce the large anomaly-contaminated unlabeled data $\mathcal{D}^u$. We then create 13 datasets, with each dataset having $\mathcal{D}^a$ sampled from only *one specific anomaly class*; key statistics of these datasets are shown in Table 5, where each dataset is named by the known anomaly class. The test data is fixed after the training-test data split, which contains one known and one-to-six unknown anomaly classes. The anomalies account for 0.96%-5.78% of the test data.

Table 5: Key statistics of 13 datasets created from the four data bases: *NB15*, *Thyroid*, *HAR* and *Covertype*.

Data Base	Dataset	Data Size	D	Anomaly Class	Normal Class	Anomaly Proportion
NB15	Analysis	95,677	196	Analysis	Normal network flows	2.80%
	Backdoor	95,329	196	Backdoor		2.44%
	DoS	96,000	196	DoS		3.13%
	Exploits	96,000	196	Exploits		3.13%
	Fuzzers	96,000	196	Fuzzers		3.13%
	Generic	96,000	196	Generic		3.13%
	Recon	96,000	196	Recon		3.13%
Thyroid	Hypothyroid	6,832	21	Hypothyroid	Normal patients	2.43%
	Subnormal	7,034	21	Subnormal		5.23%
HAR	Downstairs	7,499	561	Downstairs	Common human activities	2.00%
	Upstairs	7,499	561	Upstairs	(Classes 1, 4, 5, 6)	2.00%
Covertype	Cottonwood	286,048	54	Cottonwood	Lodgepole Pine	0.96%
	Douglas-fir	300,668	54	Douglas-fir		5.78%

C Implementation Details

Algorithm Implementation. All methods are implemented using Python, with DevNet and REPEN directly taken from the authors at https://sites.google.com/site/gspangsite/sourcecode and iForest taken from the scikit-learn package. Deep anomaly detection methods DevNet, DevNet$^+$, REPEN and Deep SAD are built upon Keras with Tensorflow as the backend. We implement DPLAN based on the deep Q-network implementation in the open-source Keras-based deep reinforcement learning project, namely, Keras-rl, available at https://github.com/keras-rl/keras-rl. Our anomaly-biased simulation environment is implemented under the OpenAI Gym environment. The main packages and their versions used in this work are provided as follows:

- gym==0.12.5
- keras==2.3.1
- keras-rl==0.4.2
- numpy==1.16.2
- pandas==0.23.4
- scikit-learn==0.20.0
- scipy==1.1.0
- tensorboard==1.14.0
- tensorflow==1.14.0

Hyperparameter Settings. The network architecture used in DPLAN, DevNet, Deep SAD and REPEN contains one hidden layer with 20 ReLU units by default. The DPLAN with a deeper network architecture (*i.e.*, the variant of DPLAN - DQN$^+$) adds two additional hidden layers immediately after the input layer. The first hidden layer contains 500 ReLU units while the second hidden layer contains 100 ReLU. Following each of these two hidden layers, we add a dropout layer to avoid overfitting. The dropout rate is 0.9 for both dropout layers.

Since original deep Q-network is designed for complex control tasks with a large set of possible actions in very high-dimensional space, some of its recommended parameter settings are not applicable to our anomaly detection task with two possible actions. Therefore, in addition to adapt the network architecture, some parameters also need to be accordingly adapted. Specifically, DPLAN is trained with 20,000 steps by default, with 10,000 warm-up steps and the target network updated every 10,000 steps. Each episode contains 2,000 steps. The episode is terminated only when the 2,000 steps are completed. We update the parameters θ^e in the intrinsic reward function f every episode (*i.e.*, 2,000 steps). Also, as shown in Algorithms 1 and 2, the ϵ greedy exploration is only used in our model training, with ϵ annealed from 1.0 to 0.1 over the course of 10,000 steps; it is not used in our evaluation since we does not need any further exploration during testing. The experience replay memory size is set to 100,000 since our agent can typically converge very early. The other parameters of deep Q-network are set to the default settings as in the original DQN [36], with some key hyperparameter settings shown in Table 6.

DevNet, REPEN and iForest are used with the settings recommended in [4, 6, 8]. DevNet$^+$ uses the same optimization settings as DevNet. We searched the minibatch size for Deep SAD using a set of

Table 6: Key default hyperparameters from original DQN

Hyperparameter	Value
minibatch size	32
discount factor γ	0.99
learning rate	0.00025
gradient momentum	0.95
min squared gradient	0.01

options $\{8, 16, 32, 64, 128, 256, 512\}$ and found the batch size 512 works best for Deep SAD. Deep SAD generally converged after 50 epochs, with 20 minibatches per epoch. So, these settings are used by default for Deep SAD.

D Tolerance w.r.t. Anomaly Contamination Rate

Experiment Settings. We also examine the effect of varying anomaly contamination rates on the performance of the detectors. We incrementally add unlabeled anomalies into the training data with an anomaly contamination factor of $n \times 2\%$ for each anomaly class, with $n \in \{1, 2, 3, 4, 5, 6\}$.

Results. The AUC-PR results of DPLAN and its five competing anomaly detectors w.r.t. increasing anomaly contamination are shown in Figure 4. The following three remarks can be made from the results. (i) Regardless of the difference in the anomaly contamination, the superiority of DPLAN over the five competing methods is consistent to the results using 2% contamination in the main text. (ii) It is interesting that DPLAN, DevNet and Deep SAD perform stably with increasing anomaly pollution factors on several datasets, *i.e.*, *Analysis*, *DoS*, *Hypothyroid*, *Downstairs* and *Douglas-fir*, while having clear downward trends on the rest of the other datasets where the supervisory signal from the the labeled anomalies may not be strong enough to tolerate the noises. (iii) REPEN and DevNet$^+$ demonstrate very irregular performance patterns. This may be due to that both of them incorporate a pseudo labeling module into its learning process, and their performance is rather sensitive to the quality of the pseudo labels. For example, REPEN and DevNet$^+$ may achieve substantially improved performance when the true positives in the pseudo labels increase; and its performance may drop badly in the opposite cases.

E Sensitivity w.r.t. Representation Dimensionality Size

Experiment Settings. We also examine the sensitivity of DPLAN w.r.t. varying representation dimensionality size in its feature layer. A set of dimensionality sizes in a large range, *i.e.*, $\{10, 20, 40, 80, 160, 320\}$, is used.

Results. The AUC-PR results of DPLAN w.r.t. the representation dimensionality size on all the 13 datasets are shown in Figure 5. In general, DPLAN performs rather stably with different dimensionality sizes across the datasets. DPLAN performs less effectively using 10 representation dimensions than using larger representation dimensions, because 10-dimensional representations are not expressive enough to capture complex relations in most of the datasets. The performance of DPLAN becomes stable using 20 representation dimensions; increasing the dimensionality size does not change the performance much thereafter. This may be due to the fact that the supervisory information that can be leveraged by DPLAN is bounded at some point; thus, increasing the dimensionality of the learned representation space does not further improve the detection performance. In some cases where more supervisory information can be leveraged for building more complex models, such as on the dataset *Subnormal*, the performance of DPLAN is continuously improved with increasing representation dimensionality. This is consistent to the results presented in Figure 3 in the main text.

References

[1] Charu C Aggarwal. *Outlier analysis*. Springer, 2017.

[2] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15, 2009.

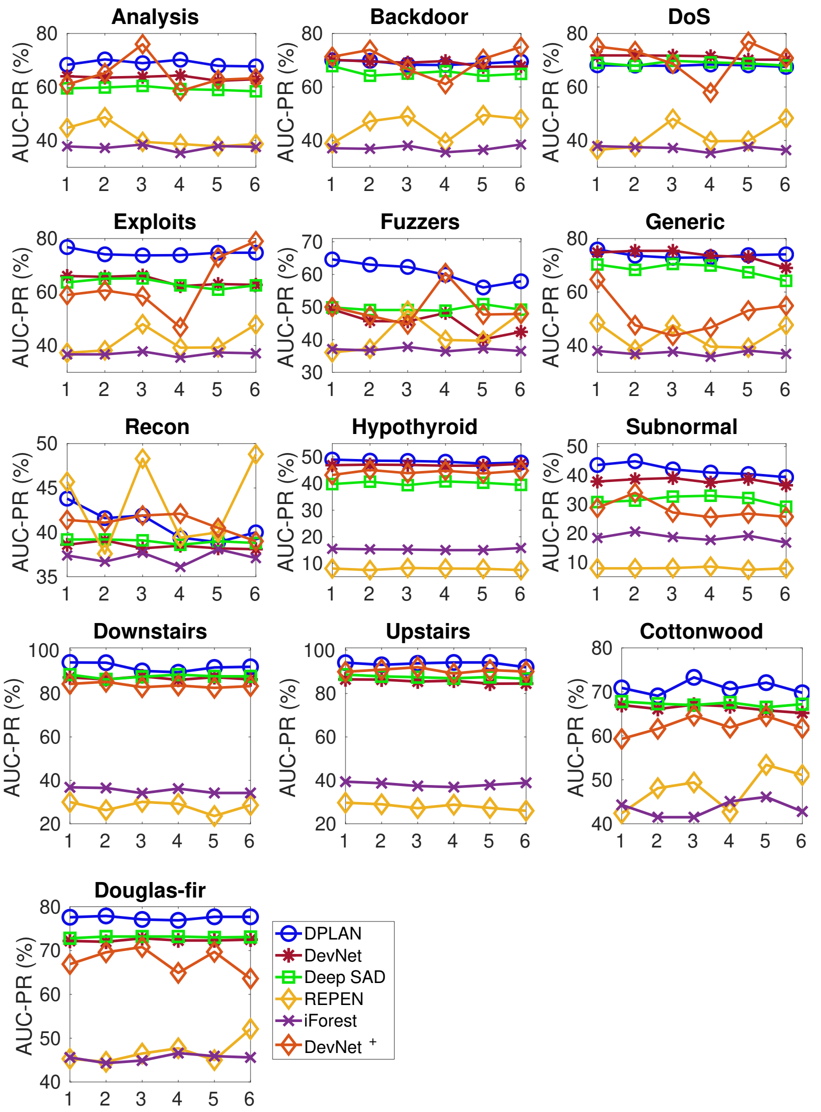

Figure 4: AUC-PR results w.r.t. different anomaly contamination factors

[3] Acar Tamersoy, Kevin Roundy, and Duen Horng Chau. Guilt by association: Large scale malware detection by mining file-relation graphs. In *KDD*, pages 1524–1533, 2014.

[4] Guansong Pang, Longbing Cao, Ling Chen, and Huan Liu. Learning representations of ultrahigh-dimensional data for random distance-based outlier detection. In *KDD*, pages 2041–2050, 2018.

[5] Ya-Lin Zhang, Longfei Li, Jun Zhou, Xiaolong Li, and Zhi-Hua Zhou. Anomaly detection with partially observed anomalies. In *WWW Workshops*, pages 639–646, 2018.

[6] Guansong Pang, Chunhua Shen, and Anton van den Hengel. Deep anomaly detection with deviation networks. In *KDD*, pages 353–362. ACM, 2019.

[7] Lukas Ruff, Robert A Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. Deep semi-supervised anomaly detection. In *ICLR*, 2020.

[8] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data*, 6(1):3, 2012.

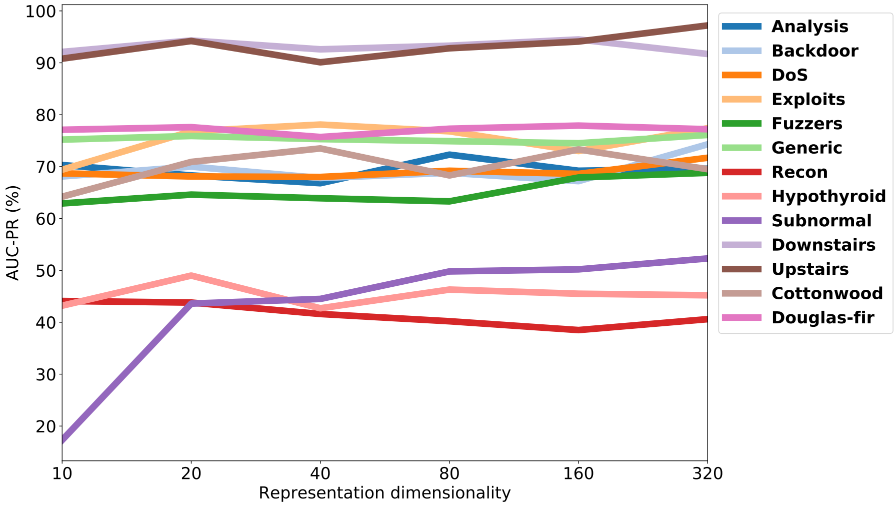

Figure 5: AUC-PR performance of DPLAN w.r.t. different representation dimensionality sizes

[9] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. LOF: Identifying density-based local outliers. *ACM Sigmod Record*, 29(2):93–104, 2000.

[10] Mahito Sugiyama and Karsten Borgwardt. Rapid distance-based outlier detection via sampling. In *NeurIPS*, pages 467–475, 2013.

[11] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *ICLR*, 2018.

[12] Guansong Pang, Anton van den Hengel, and Chunhua Shen. Weakly-supervised deep anomaly detection with pairwise relation learning. *CoRR abs/1910.13601*, 2019.

[13] Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *KDD*, pages 665–674. ACM, 2017.

[14] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga. Outlier detection with autoencoder ensembles. In *SDM*, pages 90–98. SIAM, 2017.

[15] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *IPMI*, pages 146–157. Springer, Cham, 2017.

[16] Rithesh Kumar, Sherjil Ozair, Anirudh Goyal, Aaron Courville, and Yoshua Bengio. Maximum entropy generators for energy-based models. *CoRR abs/1901.08508*, 2019.

[17] Houssam Zenati, Manon Romain, Chuan-Sheng Foo, Bruno Lecouat, and Vijay Chandrasekhar. Adversarially learned anomaly detection. In *ICDM*, pages 727–736. IEEE, 2018.

[18] Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. GANomaly: Semi-supervised anomaly detection via adversarial training. In *ACCV*, pages 622–637. Springer, 2018.

[19] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. In *NeurIPS*, pages 9758–9769, 2018.

[20] Siqi Wang, Yijie Zeng, Xinwang Liu, En Zhu, Jianping Yin, Chuanfu Xu, and Marius Kloft. Effective end-to-end unsupervised outlier detection via inlier priority of discriminative network. In *NeurIPS*, pages 5960–5973, 2019.

[21] Lukas Ruff, Nico Görnitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Robert Vandermeulen, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *ICML*, pages 4390–4399, 2018.

[22] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. Adversarially learned one-class classifier for novelty detection. In *CVPR*, pages 3379–3388, 2018.

[23] Pramuditha Perera, Ramesh Nallapati, and Bing Xiang. OCGAN: One-class novelty detection using gans with constrained latent representations. In *CVPR*, pages 2898–2906, 2019.

[24] Panpan Zheng, Shuhan Yuan, Xintao Wu, Jun Li, and Aidong Lu. One-class adversarial nets for fraud detection. In *AAAI*, volume 33, pages 1286–1293, 2019.

[25] Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. Future frame prediction for anomaly detection–a new baseline. In *CVPR*, pages 6536–6545, 2018.

[26] Davide Abati, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Latent space autoregression for novelty detection. In *CVPR*, pages 481–490, 2019.

[27] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton van den Hengel. Deep learning for anomaly detection: A review. *arXiv preprint arXiv:2007.02500*, 2020.

[28] Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *IJCAI*, pages 587–592, 2003.

[29] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *KDD*, pages 213–220. ACM, 2008.

[30] Gang Niu, Marthinus Christoffel du Plessis, Tomoya Sakai, Yao Ma, and Masashi Sugiyama. Theoretical comparisons of positive-unlabeled learning against positive-negative learning. In *NeurIPS*, pages 1199–1207, 2016.

[31] Emanuele Sansone, Francesco GB De Natale, and Zhi-Hua Zhou. Efficient training for positive unlabeled learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[32] Dan Pelleg and Andrew W Moore. Active learning for anomaly and rare-category detection. In *NeurIPS*, pages 1073–1080, 2005.

[33] Naoki Abe, Bianca Zadrozny, and John Langford. Outlier detection by active learning. In *KDD*, pages 504–509, 2006.

[34] Shubhomoy Das, Weng-Keen Wong, Thomas Dietterich, Alan Fern, and Andrew Emmott. Incorporating expert feedback into active anomaly discovery. In *ICDM*, pages 853–858. IEEE, 2016.

[35] Md Amran Siddiqui, Alan Fern, Thomas G Dietterich, Ryan Wright, Alec Theriault, and David W Archer. Feedback-guided anomaly discovery via online optimization. In *KDD*, pages 2200–2209, 2018.

[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[37] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[38] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[39] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. In *KDD*, pages 1040–1048, 2018.

[40] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. DRN: A deep reinforcement learning framework for news recommendation. In *WWW*, pages 167–176, 2018.

[41] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

[42] Xingping Dong, Jianbing Shen, Wenguan Wang, Yu Liu, Ling Shao, and Fatih Porikli. Hyperparameter optimization for tracking with continuous deep q-learning. In *CVPR*, pages 518–527, 2018.

[43] Min-hwan Oh and Garud Iyengar. Sequential anomaly detection using inverse reinforcement learning. In *KDD*, pages 1480–1490, 2019.

[44] Andrew Y Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *ICML*. Citeseer, 2000.

[45] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, pages 2778–2787, 2017.

[46] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. In *ICLR*, 2019.

[47] Nour Moustafa and Jill Slay. UNSW-NB15: a comprehensive data set for network intrusion detection systems. In *Military Communications and Information Systems Conference, 2015*, pages 1–6, 2015.

[48] Md Amran Siddiqui, Alan Fern, Thomas G. Dietterich, Ryan Wright, Alec Theriault, and David W. Archer. Feedback-guided anomaly discovery via online optimization. In *KDD*, pages 2200–2209. ACM, 2018.

[49] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra Perez, and Jorge Luis Reyes Ortiz. A public domain dataset for human activity recognition using smartphones. In *ESANN*, pages 437–442, 2013.

[50] Kai Ming Ting, Takashi Washio, Jonathan R Wells, and Sunil Aryal. Defying the gravity of learning curve: a characteristic of nearest neighbour anomaly detectors. *Machine Learning*, 106(1):55–91, 2017.

[51] Jock A Blackard. *Comparison of neural networks and discriminant analysis in predicting forest cover types.* PhD thesis, Department of Forest Sciences. Colorado State University, 2000.

[52] Leman Akoglu, Hanghang Tong, Jilles Vreeken, and Christos Faloutsos. Fast and reliable anomaly detection in categorical data. In *CIKM*, pages 415–424, 2012.

[53] David MJ Tax and Robert PW Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

[54] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[55] Kendrick Boyd, Kevin H Eng, and C David Page. Area under the precision-recall curve: point estimates and confidence intervals. In *ECML/PKDD*, pages 451–466. Springer, 2013.

[56] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan):1–30, 2006.

[57] Sam Corbett-Davies and Sharad Goel. The measure and mismeasure of fairness: A critical review of fair machine learning. *CoRR abs/1808.00023*, 2018.

[58] James E Johndrow, Kristian Lum, et al. An algorithm for removing sensitive information: application to race-independent recidivism prediction. *The Annals of Applied Statistics*, 13(1):189–220, 2019.

[59] Nguyen Xuan Vinh, Jeffrey Chan, Simone Romano, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Jian Pei. Discovering outlying aspects in large datasets. *Data Mining and Knowledge Discovery*, 30(6):1520–1555, 2016.

[60] Md Amran Siddiqui, Alan Fern, Thomas G Dietterich, and Weng-Keen Wong. Sequential feature explanations for anomaly detection. *ACM Transactions on Knowledge Discovery from Data*, 13(1):1–22, 2019.

[61] Guilherme O Campos, Arthur Zimek, Jörg Sander, Ricardo JGB Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E Houle. On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30(4):891–927, 2016.