Enhancing Robustness of On-line Learning Models on Highly Noisy Data

Zhao, Zilong; Birke, Robert; Han, Rui; Robu, Bogdan; Bouchenak, Sara; Ben Mokhtar, Sonia; Chen, Lydia Y.

DOI
10.1109/TDSC.2021.3063947

Publication date
2021

Document Version
Final published version

Published in
IEEE Transactions on Dependable and Secure Computing

Enhancing Robustness of On-Line Learning Models on Highly Noisy Data

Zilong Zhao, Robert Birke, *Senior Member, IEEE*, Rui Han, Bogdan Robu, Sara Bouchenak, Sonia Ben Mokhtar, and Lydia Y. Chen, *Senior Member, IEEE*

Abstract—Classification algorithms have been widely adopted to detect anomalies for various systems, e.g., IoT, cloud and face recognition, under the common assumption that the data source is clean, i.e., features and labels are correctly set. However, data collected from the wild can be unreliable due to careless annotations or malicious data transformation for incorrect anomaly detection. In this article, we extend a two-layer on-line data selection framework: Robust Anomaly Detector (RAD) with a newly designed ensemble prediction where both layers contribute to the final anomaly detection decision. To adapt to the on-line nature of anomaly detection, we consider additional features of conflicting opinions of classifiers, repetitive cleaning, and oracle knowledge. We on-line learn from incoming data streams and continuously cleanse the data, so as to adapt to the increasing learning capacity from the larger accumulated data set. Moreover, we explore the concept of oracle learning that provides additional information of true labels for difficult data points. We specifically focus on three use cases, (i) detecting 10 classes of IoT attacks, (ii) predicting 4 classes of task failures of big data jobs, and (iii) recognising 100 celebrities faces. Our evaluation results show that RAD can robustly improve the accuracy of anomaly detection, to reach up to 98.95 percent for IoT device attacks (i.e., +7%), up to 85.03 percent for cloud task failures (i.e., +14%) under 40 percent label noise, and for its extension, it can reach up to 77.51 percent for face recognition (i.e., +39%) under 30 percent label noise. The proposed RAD and its extensions are general and can be applied to different anomaly detection algorithms.

Index Terms—Unreliable data, anomaly detection, failures, attacks, machine learning

1 Introduction

Anomaly detection is one of the core operations for enforcing dependability and performance in modern distributed systems [35], [51]. Anomalies can take various forms including erroneous data produced by a corrupted IoT device or the failure of a job executed in a datacenter [6], [7], [54].

Dealing with this issue has often been done in recent art by relying on machine learning-based classification algorithms over system logs [12], [15] or backend collected data [21], [53]. These systems often rely on the assumption of clean datasets from which the classifier learns to distinguish between data corresponding to a correct execution of the system from data corresponding to an abnormal execution of the latter (i.e., anomaly detection). As workloads at real systems are highly dynamic over time, it is even more challenging to predict anomalies that can not be easily distinguished from the system dynamics, compared to the systems with static workloads.

- *Zilong Zhao and Bogdan Robu are with the Université Grenoble Alpes, 38400 Saint-Martin-d'Hères, France. E-mail: {zilong.zhao, bogdan.robu}@gipsa-lab.fr.*
- *Robert Birke is with the ABB Research, 5405 Baden, Switzerland. E-mail: robert.birke@ch.abb.com.*
- *Rui Han is with the Beijing Institute of Technology, Beijing 100811, China. E-mail: hanrui@bit.edu.cn.*
- *Sara Bouchenak and Sonia Ben Mokhtar are with the INSA Lyon, 69100 Villeurbanne, France. E-mail: {sara.bouchenak, sonia.benmokhtar}@insa-lyon.fr.*
- *Lydia Y. Chen is with the TU Delft, 2628 CD, Delft, The Netherlands. E-mail: y.chen-10@tudelft.nl.*

Manuscript received 30 Sept. 2019; revised 18 Jan. 2021; accepted 21 Feb. 2021. Date of publication 4 Mar. 2021; date of current version 1 Sept. 2021. (Corresponding author: Rui Han.)
Digital Object Identifier no. 10.1109/TDSC.2021.3063947

In this context, a rising concern when applying classification algorithms is the accessibility to a reliable ground truth for anomalies [10]. Typically, anomaly data is manually annotated by human experts and hence the generation of anomaly labels is subject to quality variation, so-called noisy labels. For instance, annotating service failure types for data centers is done by operators of varying expertise.

However, standard machine learning algorithms typically assume clean labels and overlook the risk of noisy labels. Moreover, recent studies point out the increase in dirty data attacks that can maliciously alter the anomaly labels to mislead the machine learning models [11], [14], [19], [23]. As a result, anomaly detection algorithms need to capture not only anomalies that are entangled with system dynamics but also the unreliable nature of anomaly labels.

Indeed, a strong anomaly classification model can be learned by incorporating a larger amount of data. However, learning from data with noisy labels can significantly degrade the classification accuracy, even for deep neural networks [14], [46], [52]. Such concerns lead us to ask the following question: how to build an anomaly detection framework that can robustly differentiate between true and noisy anomalies and efficiently learn anomaly classification models from a succinct amount of clean data. The immediate challenge of capturing the data quality lies at the fact that label qualities are not directly observable but only via anomaly classification outcomes that in turn are coupled with the noise level in data labels.

We extend Robust Anomaly Detector (RAD) [54], a generic framework that continuously learns an anomaly classification model from streams of event logs or images that are subject to label noise. The original design of RAD is

composed of two layers of learning models, i.e., a data label model and an anomaly classifier. The label model aims at differentiating the label quality, i.e., noisy versus true labels, for each batch of new data and only "clean" data points are fed in the anomaly classifier. The anomaly classifier predicts the event outcome that can be divided in multiple classes of (non)anomalies, depending on the specific use case. In this extension, we derive three alternatives of RAD, namely, voting, active learning and slim. These use additional information, e.g., opinions of conflicting classifiers and queries of oracles. We iteratively update the prediction of historical windows such that weak predictions can be continuously improved by the latest model. Moreover, we propose an ensemble prediction strategy to reconcile the prediction outcomes of the two models, namely label model and anomaly classification model, instead of only relying on classification model as [54]

To demonstrate the effectiveness of RAD, we consider three use cases using open datasets: detecting 10 classes of attacks on IoT devices [28], predicting 4 types of task failures for big data processing cluster [37], [40], and recognising 100 most abundant celebrity faces [31]. Our results show that RAD can effectively cleanse the data, i.e., selecting data with clean labels, and result in better anomaly detection accuracy per additional included streamed data, compared to classifiers without continuous data cleansing. Specifically, under 40 percent noise, RAD achieves up to 98.95 percent, 85.03 percent (comparing to 92.27 and 71.02 percent by anomaly classification model of no selection on dataset) for detecting IoT device attacks and predicting cluster task failures, respectively. If we implement RAD Active Learning on cluster dataset with the same noise level, the final accuracy could improve from 85.03 to 90.77 percent. For face image dataset, final accuracy of RAD Slim under 30 percent noise achieves to 77.51 percent (comparing to 38.89 percent of no selection on dataset). Furthermore, our study shows that RAD is stable even when the noise is very strong. And if we do not have many clean data at beginning to pre-train the model, RAD Active Learning and RAD Active Learning Limited can still perform very well from a very bad starting model.

The main contributions of this study can be summarized as follows:

- We design an effective on-line anomaly detection framework, RAD, consisting of a data selection and prediction module that cater to a wide range of implementation choices from regular machine learning models to deep neural networks.
- We explore three novel data selection schemes: namely voting, active learning, and active learning limit. These can filter out the suspicious data and call upon experts to cleanse the data based on the predicted uncertainty from the quality model and classification model. We combine the novel ideas of model disagreement and active learning.
- We leverage the power of ensemble model prediction to enhance the robustness of trained anomaly classifier by incorporating the predictions of the label model used in the data selection.
- RAD can be applied on multiple types of anomaly inputs, i.e., server failure, IoT devices, and images. Specially, RAD Active Learning can achieve remarkable accuracy similar to the performance under no label noise.

The remainder of the paper is organized as follows. Section 3 describes the motivating case studies. Sections 4 and 5 present the proposed RAD framework and the results of its experimental evaluation, respectively. Section 2 describes the related work. Finally, Section 6 draws our conclusions.

2 Related Work

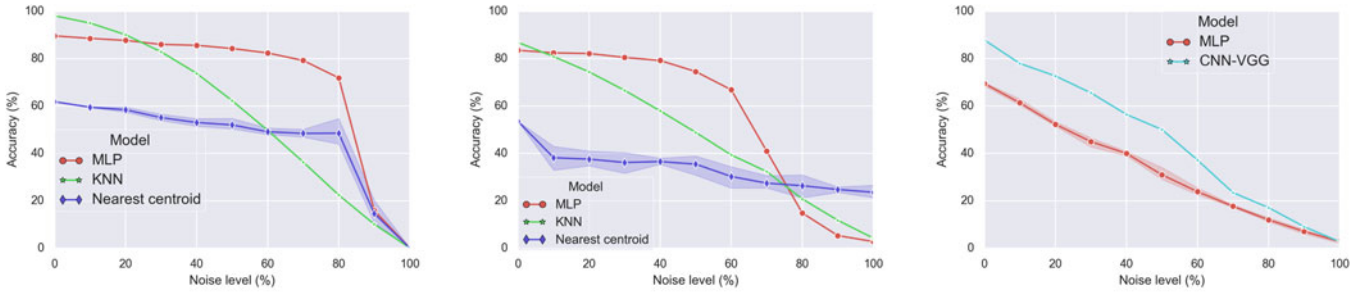
Machine learning has been extensively used for failure detection [9], [34], [36], [38], attack prediction [1], [3], [4], [24], [25], [57], and face recognition [41], [44], [49]. Considering noisy labels in classification algorithms is also a problem that has been explored in the machine learning community as discussed in [5], [13], [30].

The problem of classification in presence of noisy labels can be organized into various categories according to, on the one hand, the type of classification algorithm subject to noise, and on the other hand, the techniques used to remove the noise.

Noisy labels have been studied for binary classification where noisy labels are considered as symmetric (e.g., [26]) and for classification with multiple classes where noisy labels are considered as asymmetric, e.g., [32], [43]. For this paper, we consider the problem of classification with multiple classes. Furthermore, noisy labels have been considered in various types of classifiers KNN [50], SVM [2], and deep neural networks [47]. In the context of this paper, our proposed approach is agnostic to the underlying classifier type as noise removal is performed ahead of the classification.

Various techniques explore countering label noise following two main strategies. The first strategy trains a single model as filter for noisy label data. [1], [20], [27], [48] train a separate filter from clean data for distinguishing noisy labels. Instead [45] trains on the original data (with noise). Training the filter with clean data is better, but the assumption of large quantity of clean data does not always hold, especially in our on-line learning scenario. Using noisy data to train a filter raises a *chicken-and-egg* dilemma [13], since: 1) good classifiers are necessary for filtering but 2) learning from noisy label data may precisely produce poor classifiers.

The second strategy relies on voting based algorithms to mitigate possible biases stemming from a chosen single filter. [8], [22], [29] simultaneously train several classifiers directly on the original data. Afterwards, they use either majority vote (i.e., classify a sample as mislabeled if a majority of classifiers misclassified it) or consensus vote (i.e., classify a sample as mislabeled if all classifiers misclassified the sample) to filter noisy data. There are similarities between these algorithms and our RAD Voting and RAD Active Learning. However, these solutions focus on static datasets and the off-line setting. They do not consider the learning efficiency and training limitation for on-line scenarios. Since the interval between data batches can be short, we need to ensure the training and inference times per batch are as short as possible. The two models in our RAD are connected in cascade. Only data instances deemed uncertain by the first model get to be predicted twice. Off-line voting methods instead need to process each data record multiple times,



(a) Use case of IoT thermostat device attacks (b) Use case of Cluster task failures (c) Use case of Face Recognition

Fig. 1. Impact of noisy data on anomaly classification.

once for each classifier trained by the algorithm. [22] trains two neural networks on top of the classification model. In image classification, training three big CNNs can be hardware-impossible on many devices or very slow and not suitable for on-line learning.

Using active learning in data cleaning has been explored in pattern recognition research. [16] proposes to define an information criteria function for patterns (data instances). If the information value is below a given threshold, the pattern can be used by the learning algorithm. Otherwise, the pattern is sent to a human expert for checking. The idea is similar to our active learning method, but we go one step beyond by limiting the number of expert queries and proposing an uncertainty-based ranking in RAD Active Learning Limited and RAD Slim Limited. Only the most valuable instances are thus selected to expert cleansing.

3 Motivating Case Studies

To qualitatively demonstrate the impact of noisy data on anomaly detection, we use three case studies.

- Detecting *IoT device attacks* from inspecting network traffic data collected from commercial IoT devices [28]. This dataset contains nine types of IoT devices which are subject to 10 types of attacks. Specifically, we focus on the Ecobee thermostat device that may be infected by Mirai malware and BASHLITE malware. Here we focus on the scenario of detecting and differentiating between 10 attacks. It is important to detect those attacks with high accuracy against all load conditions and data qualities.
- Predicting *task execution failures* for big data jobs running at a Google cluster [37], [39]. This trace contains a month-long job execution records from Google clusters. Each job contains multiple tasks, which can be terminated into four different states: *finish*, *fail*, *evict*, or *kill*. The last three states are considered as anomalous states. To minimise the computational resource waste due to anomalous states, it is imperative to predict the final execution state of task upon their arrivals.
- Recognizing *celebrity faces* from photos of the FaceScrub dataset [31]. The set is a collection of photos of celebrities roughly half female and half male. The task is to recognize faces by matching each photo to the identity of the celebrity shown on it. Here we focus on the face recognition of the 100 celebrities with the highest number of photos in the dataset totalling to 12K images. Faces are widely used in biometric identification systems in many security applications, e.g., access control. This makes the robustness of such systems critical. Furthermore, this image dataset is studied because we want to show the broad applicability of our proposed framework.

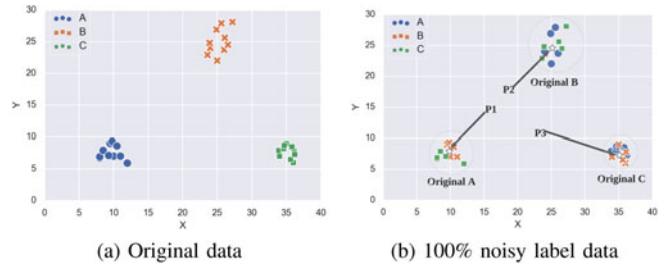
The details about data definition, and statistics, e.g., number of feature and number of data points, can be found in Section 5.1. To recognize anomalies/faces in each use case, related studies have applied different machine learning classification algorithms, from simple ones, e.g., k-nearest neighbour (KNN), to complex ones, e.g., deep neural networks (DNN), under scenarios with different levels of symmetric label noise. Noisy labels are corrupted with equal probability to all classes except the true one. Here, we evaluate how the detection accuracy changes relative to different levels of noises. We focus on off-line scenarios where we split the data in a training set affected by label noise and a clean evaluation set. Due to our focus on resistance to noisy labels, we repeat experiments by regenerating the noise while keeping the model initialisation constant.

3.1 Anomaly Detection

Classification models are learned from 14,000 training records and evaluated on a clean testing set of 6,000 records. We specifically apply KNN, nearest centroid and multilayer perceptron (MLP) (a.k.a feed-forward deep neural networks) on both the IoT device attacks and the cluster task failures. We repeat all experiments 10 times.[1] Figs. 1a and 1b summarize the accuracy results.

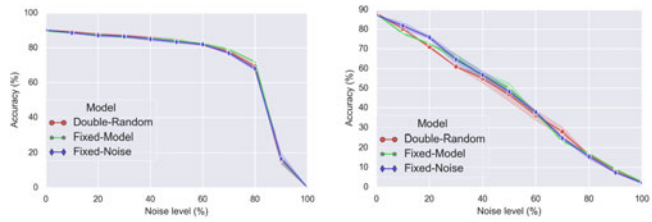
One can see that noisy labels clearly deteriorate the detection results for both IoT attacks and task failures, across all three classification algorithms. For standard classifiers, like KNN and nearest centroid, the detection accuracy decays faster than MLP which is more robust to noisy labels. Such an observation holds for both use cases. For IoT attacks, MLP can even achieve a similar accuracy as the no label noise case, when 40 percent of label classes are altered. Moreover, the impact of noise depends also heavily on the specific sequence of label noise. Corrupting the labels of some samples has a higher impact than corrupting others. As a consequence the curve is highly unstable with large variances and leads sometimes to counter intuitive results

1. To verify and reproduce the results the code is available at https://github.com/zhao-zilong/MotivationCaseStudies



(a) Original data (b) 100% noisy label data

Fig. 2. Example of training under 100 percent label noise.



(a) IoT attack (b) Face recognition

Fig. 3. Impact of model versus noise randomness across runs.

of non monotonic impact of noise level on accuracy. An example is given by the nearest centroid results on the cluster task dataset. Even across 100 runs the mean accuracy at 30 percent noise is slightly lower than the mean accuracy at 40 percent noise.

3.2 Face Recognition

For face recognition we use a subset of our complete dataset (which contains 100 celebrities). The subset contains 2,639 images from 20 celebrities with varying degrees of label noise as training set and 665 clean images as testing set. Due to the complex features of image data, we use a MLP and convolutional neural networks (CNN). Specifically we use a small VGG [42] with 6 convolutional layers. We repeat MLP experiments 10 times, and VGG experiments 3 times due to the higher training complexity.[1] Fig. 1c shows the accuracy results under different label noise levels. Similar to previous use cases, one can observe that label noise strongly affects the performance of both classifiers. The accuracy degradation is approximately linear with the noise level. VGG outperforms MLP in this dataset under all noise levels except 100 percent corrupted labels.

Although it is rare to encounter a dataset with 100 percent noisy labels, it is still an interesting scenario to study. Almost all accuracy curves in Fig. 1 reach near 0 percent under 100 percent noise. This can be counter intuitive as illustrated by the following example. If the dataset contains K balanced classes, one might think that it should be possible to obtain an accuracy of $\frac{1}{K}$ just by guessing. However training on 100 percent noisy label data is worse than random guessing. We illustrate this via a simple example with three classes, A, B and C, and 10 samples per class. Fig. 2a shows the original sample distribution. Fig. 2b shows the sample distribution with 100 percent label noise. Since all labels are corrupted, each original cluster only contains labels of wrong classes. If we train a machine learning model (e.g., KNN with $k = 5$) on this noisy label data, we learn a wrong model which can misclassify any data point. See the highlighted points P1, P2 and P3 in Fig. 2b as examples. Training on 100 percent noisy label data is hence worse than zero-knowledge guessing because fully corrupted data can mislead the learning process.

We fix the model initialization and regenerate the noisy data across experiment runs. The randomness of the results only stems from the noise injected into the data labels used for training. Before choosing this setting, we run preliminary experiments with different types of randomness. Fig. 3 shows the results on the IoT attack and Face recognition datasets using MLP and VGG, respectively. It compares: *Fixed-Model* initialization with regenerated noise – our setting throughout the paper; *Fixed-Noise* with random model initialization; and *Double-Random* with regenerated noise and random model initialization. Both cases show significant overlaps between the three types of randomness, especially for MLP. Due to the lower number of runs (3 against 10) and higher model complexity, the VGG results are slightly more dispersed. Neither case shows significant impact of the randomness type on the results. As our study focuses on the influence of noisy label data, we choose *Fixed-Model* for the remainder of the paper.

The above three experiments clearly show that under the presence of noisy label data, all models are progressively degraded. These cases motivate us to design the RAD framework and its extension to counter the influence of noisy label data on the learning process.

4 Design Principles of RAD Framework

In this section, we introduce the system model followed by the general structure of RAD and its extended features with respect to data selection and model prediction – ensemble prediction. All used symbols are summarized in Table 1.

4.1 System Model

We consider a dataset that consists of several data instances. Each data instance has f features. Each data instance belongs to a class k, where $k \in \mathcal{K} = \{1, \ldots, K\}$. Data instances are part of a pre-labeled dataset $\mathcal{D}$ with labels Y used for training. Furthermore, a labeled data instance is either correctly labeled (i.e., clean data instance), or incorrectly labeled (i.e., noisy data instance). We use the indicator variable $\hat{q}$ to indicate clean $\hat{q} = 1$ and dirty $\hat{q} = 0$ labels. Wrong labels can stem from several reasons ranging from subjectivity and data-entry errors, to malicious error injection. The quality of a dataset $\mathcal{D}$ is measured as the percent of clean labeled data instances, denoted here as $\tilde{Q}$.

Data instances arrive at the learning system continuously over time in batches. $\mathcal{D}_i$ denotes the batch of labeled data arriving at time t_i and having labels Y_i. In general we denote the time window with the subscript i. We assume that a small initial batch of data instances $\mathcal{D}_0$ has only clean labels, that is $\tilde{Q}_0 = 100\%$. Subsequent batches, include varying proportions of noisy labels, i.e $0 < \tilde{Q}_i < 100\%, i > 0$. For simplicity we consider arriving batches of equal size, $\forall \mathcal{D}_i, |\mathcal{D}_i| = N$, but not necessarily at regular times.

A classification request consists of a batch of non-labeled data instances $\mathcal{P}_i$ for which the classifier predicts the class k of each data instance. At each batch arrival, the classification output $\hat{Y}_i$ is thus an array of the predicted classes for each non-labeled data instance.

TABLE 1
Symbol Description

Symbol	Description
$\mathfrak{L}$	label quality predictor
$\mathfrak{C}$	anomaly detection classifier
$\mathcal{D}_i$	ith training data batch
$\mathcal{D}_i^*$	ith cleansed data batch from $\mathfrak{L}$
$\mathcal{P}_i$	ith test data batch
$\hat{Y}_i$	prediction of ith test data batch from $\mathfrak{C}$
$\tilde{Q}_i$	percent of clean labeled data of ith batch
$\mathcal{U}_i$	"unclean" data of ith batch determined by $\mathfrak{L}$
$\mathcal{U}_i^*$	ith cleansed data batch from $\mathfrak{C}$
$\mathcal{S}_i$	"unclean" data of ith batch determined by $\mathfrak{C}$
$\mathcal{S}_i^*$	data with true label from Expert of ith batch
$\hat{p}$	indicator of prediction, 1 for clean, 0 for dirty
$\hat{q}$	indicator of prediction, 1 for clean, 0 for dirty
α	accuracy on testing set

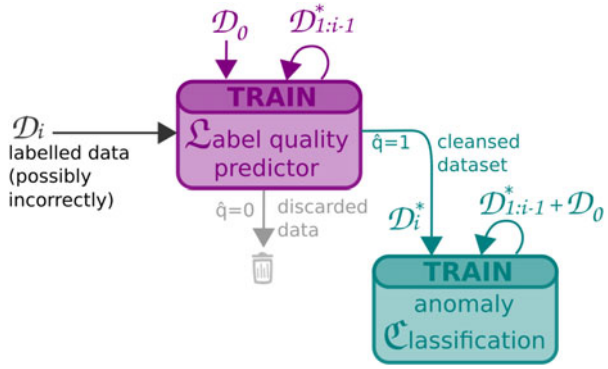


Fig. 5. RAD training data selection framework. Each block is a machine learning algorithm. Data used to train is represented by colored arrows from the top. The flowchart is iterated at every batch arrival with new labelled and unlabelled data coming in (black arrows on the left). The labelled training data for $\mathfrak{C}$ is cleansed based on the label quality predicted by $\mathfrak{L}$.

4.2 Design Overview of RAD

We propose the RAD learning framework. Its objective is threefold:

1) Learn accurate models from noisy data.
2) Continuously update the learned models based on new incoming data.
3) Propose a general approach that fits to different machine learning algorithms and different application use cases.

RAD is composed of two key steps: training data selection and class prediction, as shown in Fig. 4. Training data selection focuses on how to filter out suspicious noisy data instances and solicit *clean* data to subsequently train the classification model. It has four options: *basic*, *voting*, *active* and *slim*. The class prediction uses different prediction techniques. Available options are *ensemble*, which combines the prediction outcomes of quality and classification models; and *slim*, which has only one model to filter and classify anomalous images. We consider the following specific combinations: (i) *basic*, *voting*, and *active* are followed by the *ensemble* prediction; (ii) *slim* is followed by the *slim* prediction, which only uses one model to save computation resources.

Fig. 5 describes the overall architecture of RAD training data selection. it comprises two main components. A label quality model $\mathfrak{L}$ mainly aims at discerning clean labels from dirty labels and a classifier model $\mathfrak{C}$ targets the specific classification task at hand. But both models are used for the ensemble predictions, described in Section 4.2.3.

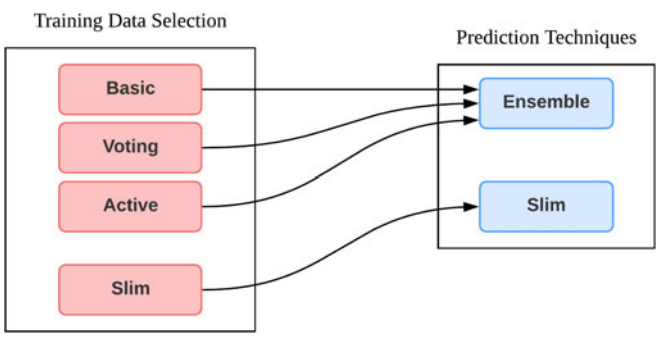


Fig. 4. Structures of RAD and its extensions: four choices of data selection and two choices of class prediction.

RAD follows a generic approach since the proposed classification framework can be used with any supervised machine learning algorithm, such as SVM, KNN, random forest, nearest centroid, DNN, etc. Moreover, RAD can be applied to a large spectrum of different applications where noisy data are collected and must be cleansed before used to train the classification model. Examples are the failure detection, attack diagnosis and face recognition illustrated in Section 5.

4.2.1 Data Selection Scheme

The first component of RAD aims to select clean data instances from $\mathcal{D}$ through the quality model. The objective of the label quality model is to select the most representative data instances to train a strong classifier model. It solicits data instances with clean labels, avoiding the pitfall that the classifier overfits to the noise. RAD uses supervised-learning algorithms to continuously train the label quality model from accumulated predicted clean data instances, to build a strong classifier.

We term the following selection procedure as *basic*, that is the default data selection scheme of RAD which requires no addition history data lookup nor involvement of human experts. $\mathfrak{L}_{i-1}$ is the label quality model that is trained with data instances received up to time $t_i - 1$, that is $\mathcal{D}_0 \ldots \mathcal{D}_{i-1}$. Upon the arrival of a new batch of data instances $\mathcal{D}_i$ at time t_i, we use the currently learned label quality model $\mathfrak{L}_{i-1}$ to predict the label quality $\hat{q}$ for each data instance in $\mathcal{D}_i$ by comparing the given k and predicted class $\hat{k}^{\mathfrak{L}_i}$. If they coincide, we consider the label as clean $q = 1$, otherwise as dirty $q = 0$. Then we build $\mathcal{D}_i^*$ as the subset of data instances from $\mathcal{D}_i$ with $q = 1$ and discard the instances with $q = 0$. This data flow is summarized in Algorithm 2.

4.2.2 Generic Approach to Handle Dynamic Data

The second component of RAD is the data classifier $\mathfrak{C}$, whose input data has dynamic noise ratios. $\mathfrak{C}_i$ is trained on

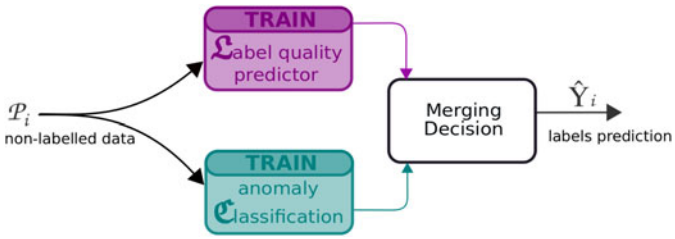


Fig. 6. Ensemble prediction.

all predicted clean data instances $\mathcal{D}^*$ received until time t_i, that is $\mathcal{D}^*{}_0 \ldots \mathcal{D}^*{}_i$. We assume that $\mathcal{D}_0$ contains only clean data instances to kick-start the framework and use the label quality model $\mathfrak{L}_0 \ldots \mathfrak{L}_{i-1}$ to cleanse $\mathcal{D}_1 \ldots \mathcal{D}_i$ and produce $\mathcal{D}^*{}_1 \ldots \mathcal{D}^*{}_i$. Thus, the RAD framework uses the batch-by-batch updated data label quality model to enrich the training data of the classification model.

4.2.3 Prediction Techniques

Fig. 6 shows the structure of ensemble prediction, which combines the prediction outcomes of both the quality and classification models. The combined decision leverages the confidence from the output probability vectors and the test accuracy of both models from the previous training epoch. If the predictions of the two models coincide, the common prediction is used. If not, we use the prediction of the model having higher confidence. As for the confidence measure, we use the class probability from the output vector multiplied by the test accuracy of the last epoch. We provide the details in Algorithm 1.

Algorithm 1. Ensemble Prediction

Input: Test data $\mathcal{P}_i$, label quality model $\mathfrak{L}_i$, classification model $\mathfrak{C}_i$, testing accuracy $\alpha^{\mathfrak{L}_{i-1}}$ of $\mathfrak{L}_{i-1}$ and $\alpha^{\mathfrak{C}_{i-1}}$ of $\mathfrak{C}_{i-1}$.

Conv(): convert probability vector to class. Max(): return maximum value in a vector.

Output: Predicted labels $\hat{Y}_i$

Get predicted labels for $\mathcal{P}_i$ by $\mathfrak{L}_i$ ($Y_i^{\mathfrak{L}^{\mathcal{P}}}$) and $\mathfrak{C}_i$ ($Y_i^{\mathfrak{C}^{\mathcal{P}}}$). Both have length $|\mathcal{P}_i|$ where each element is a vector with probabilities for each of the K classes summing to 1.
Initialize an empty $\hat{Y}_i$ of length $|\mathcal{P}_i|$
for $j \in \{1, 2, \ldots, |\mathcal{P}_i|\}$ **do**
  if Conv($Y_i^{\mathfrak{L}^{\mathcal{P}}}$[j]) = Conv($Y_i^{\mathfrak{C}^{\mathcal{P}}}$[j]) **then**
    $\hat{Y}_i$[j] ← Conv($Y_i^{\mathfrak{C}^{\mathcal{P}}}$[j])
  else
    if $\alpha^{\mathfrak{C}_{i-1}} \times Max(Y_i^{\mathfrak{C}^{\mathcal{P}}}[j]) > \alpha^{\mathfrak{L}_{i-1}} \times Max(Y_i^{\mathfrak{L}^{\mathcal{P}}}[j])$ **then**
      $\hat{Y}_i$[j] ← Conv($Y_i^{\mathfrak{C}^{\mathcal{P}}}$[j])
    else
      $\hat{Y}_i$[j] ← Conv($Y_i^{\mathfrak{L}^{\mathcal{P}}}$[j])
    end if
  end if
end for
return $\hat{Y}_i$

An alternative prediction technique is *slim* implemented in RAD Slim, which relies on one single model for both data selection and classification to save on the computation overhead. We specifically apply RAD Slim on image data that demands complex convolutional neural networks.

4.3 Extended Choices for Data Selection

In addition to the basic data selection scheme, we provide three additional schemes, namely RAD Voting, RAD Active Learning, and RAD Slim. Here, we explain their specific pitfalls and opportunities.

4.3.1 RAD Voting

The base RAD uses distinctive goals for the two models. However this approach biases the results towards the label quality model $\mathfrak{L}$. We want the classifier model $\mathfrak{C}$ to also play a role in selecting clean data instances. We do this via the voting extension shown in Fig. 7.

Algorithm 2. RAD, RAD Voting and RAD Active Learning

Input: Data batch $\mathcal{D}_i$ with given labels Y_i, label quality model $\mathfrak{L}_{i-1}$, classification model $\mathfrak{C}_{i-1}$, r reprocessed batches

Output: $\mathfrak{L}_i$, $\mathfrak{C}_i$

Predict labels $Y_i^{\mathfrak{L}}$ for $\mathcal{D}_i$ using $\mathfrak{L}_{i-1}$.
Create $\mathcal{D}_i^*$ as subset of data points in $\mathcal{D}_i$ where $Y_i[j] = Y_i^{\mathfrak{L}}[j]$ for $j \in 0, \ldots, |\mathcal{D}_i|$.
$\mathfrak{L}_{i-1}$ sends $\mathcal{D}_i^*$ to $\mathfrak{C}_{i-1}$.
if Algorithm is **RAD then**
  Retrain $\mathfrak{L}_{i-1}$ and $\mathfrak{C}_{i-1}$ on all accumulated $\mathcal{D}_t^*$ t ∈ [0,i] to get $\mathfrak{L}_i$ and $\mathfrak{C}_i$
  return $\mathfrak{L}_i$, $\mathfrak{C}_i$
end if
if Algorithm is **RAD Voting** or **Active Learning then**
  Create $\mathcal{U}_i$ as subset of data points in $\mathcal{D}_i$ where $Y_i[j] \neq Y_i^{\mathfrak{L}}[j]$ for $j \in 0, \ldots, |Y_i|$.
  $\mathfrak{L}_{i-1}$ sends $\mathcal{U}_i$ (with given label Y_i^U) and predictions $Y_i^{\mathfrak{L}^U}$ to $\mathfrak{C}_{i-1}$.
  Predict labels $Y_i^{\mathfrak{C}^U}$ for $\mathcal{U}_i$ using $\mathfrak{C}_{i-1}$.
  if Algorithm is **RAD Voting then**
    Create $\mathcal{U}_i^*$ as subset of data points in $\mathcal{U}_i$ where:
    for $j \in 0, \ldots, |U_i|$:
      i) $Y_i^U[j] = Y_i^{\mathfrak{C}^U}[j]$, or
      ii) $Y_i^{\mathfrak{C}^U}[j] = Y_i^{\mathfrak{L}^U}[j]$, update $Y_i^U[j] \leftarrow Y_i^{\mathfrak{C}^U}[j]$.
    $\mathfrak{C}_{i-1}$ sends $\mathcal{U}_i^*$ to $\mathfrak{L}_{i-1}$.
    Create $\mathcal{S}_i \leftarrow \mathcal{U}_i \setminus \mathcal{U}_i^*$.
    Add $\mathcal{S}_i$ to inactive data list $L_{inac}[i]$.
    Select set r batches with highest $|L_{inac}|$. Save indexes in **R**.
    Repeat steps 1-16 for each batch with index h in **R**. Use them to update $\mathcal{D}_h *$, $\mathcal{U}_h^*$, and $\mathcal{S}_h$.
    Retrain $\mathfrak{L}_{i-1}$ and $\mathfrak{C}_{i-1}$ on all accumulated $\mathcal{D}_t^*, \mathcal{U}_t^*$ $t \in [0, i]$ to get $\mathfrak{L}_i$ and $\mathfrak{C}_i$
    return $\mathfrak{L}_i$, $\mathfrak{C}_i$
  end if
  if Algorithm is **RAD Active Learning then**
    Create $\mathcal{U}_i^*$ as subset of data points in $\mathcal{U}_i$ where $Y_i^U[j] = Y_i^{\mathfrak{C}^U}[j]$ for $j \in 0, \ldots, |U_i|$.
    $\mathfrak{C}_{i-1}$ sends $\mathcal{U}_i^*$ to $\mathfrak{L}_{i-1}$.
    Create $\mathcal{S}_i \leftarrow \mathcal{U}_i \setminus \mathcal{U}_i^*$.
    Send S_i to Expert. Expert corrects labels and returns S_i^*.
    Retrain $\mathfrak{L}_{i-1}$ and $\mathfrak{C}_{i-1}$ on all accumulated $\mathcal{D}_t^*, \mathcal{U}_t^*$ and S_i^* $t \in [0, i]$ to get $\mathfrak{L}_i$ and $\mathfrak{C}_i$.
    return $\mathfrak{L}_i$, $\mathfrak{C}_i$
  end if
end if

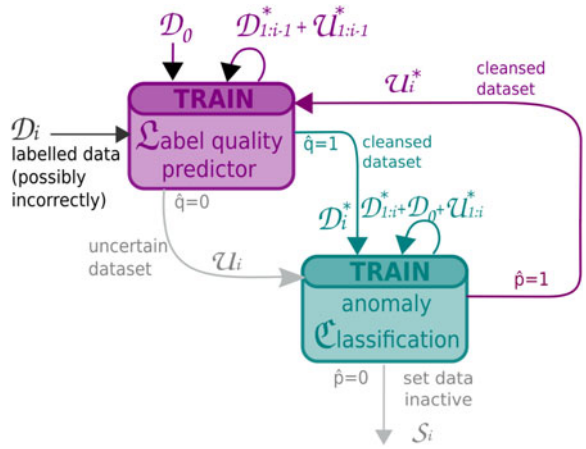


Fig. 7. RAD - Voting.

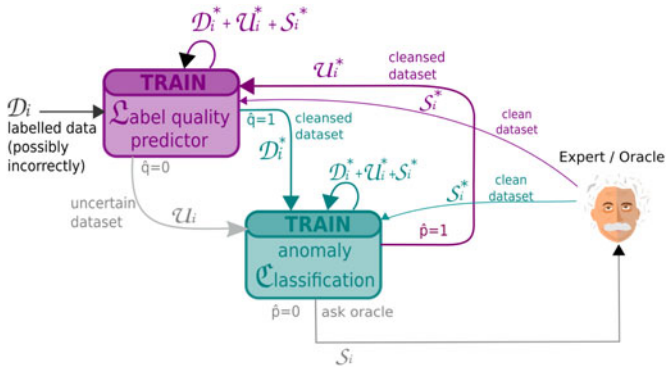


Fig. 8. RAD - Active learning.

Comparing to the base RAD, predicted dirty labels having $\hat{q} = 0$ are not discarded by $\mathfrak{L}$ but passed to $\mathfrak{C}$ as uncertain data $\mathcal{U}$. Then the classifier $\mathfrak{C}$ further cleanses the uncertain data to produce $\mathcal{U}^*$. For each data instance in $\mathcal{U}$ we predict its class $\hat{k^{\mathfrak{C}}}$ using $\mathfrak{C}$ and look for agreement with the given class k and the class $\hat{k^{\mathfrak{L}}}$ predicted by $\mathfrak{L}$. We add data instances to $\mathcal{U}^*$ if either $\hat{k^{\mathfrak{C}}}$ equals k, or if $\hat{k^{\mathfrak{C}}}$ equals $\hat{k^{\mathfrak{L}}}$. In the latter we replace the given class by the predicted class.

Batches of data instances not added to $\mathcal{U}_i^*$ at time t_i are not immediately discarded but kept in a batch $\mathcal{S}_i$ of inactive data. The idea is that since the accuracy of the classifier improves over time (see Section 5.3), we can use the new classifier to re-evaluate old batches of inactive data and further increase the training data. More in detail we maintain a list L_{inac} of the batches of inactive data $\mathcal{S}_i$. After we finish training a new classifier, we select r batches from L_{inac} with the largest number of inactive data and re-process them via the voting system. See more details in Algorithm 2. The number of batches selected from L_{inac} to re-process is a hyper-parameter. It depends on the time between data batches and the computational efficiency of the training. All training should be finished before the arrival of the next data batch. In our experiments we set $r = 2$.

4.3.2 RAD Active Learning

In RAD Voting we use $\mathfrak{C}$ and $\mathfrak{L}$ to correct labels and increase the overall amount of data used for training aiming to improve the framework accuracy. However still not all data is used. To increase further the amount of training data we resort to active learning, i.e., we ask an expert for the true class of the data instances we are least certain.

Fig. 8 shows the structure of RAD Active Learning. The difference with RAD Voting is that in RAD Active Learning we do not use the predictions from two models to correct the labels, and we do not send the most uncertain data instances to the inactive list but to an oracle to ask for the true label. In RAD Active Learning, potentially every data instance will be used to train $\mathcal{L}$ and $\mathfrak{C}$ and there is no inactive data anymore. In practice, consulting an oracle for every single uncertain data instance might be too expensive. In RAD Active Learning Limited we additionally impose a configurable limit N_{lim} on the number of data instances sent to the expert at each batch arrival. When the number of filtered out data instances exceeds N_{lim}, two lists are created: $RL_{distance}$ and RL_{std}. Both measure the uncertainty of data instances. $RL_{distance}$ ranks instances in a decreasing order based on the euclidean distance between the corresponding prediction probability vectors of $\mathfrak{C}$ and $\mathfrak{L}$. RL_{std} ranks instances in an increasing order based on the summed standard deviations of the corresponding prediction probability vectors of $\mathfrak{C}$ and $\mathfrak{L}$. We alternatively select the top instance from each list until we have N_{lim} instances. Common instances between the two lists are selected only once. We also implement experiments that only sample data from $RL_{distance}$ or RL_{std}, but the results are worse. Due to the page limit, we omit the presentation here. Current method leverages both the different opinions from the two models, and the uncertainty of each model. We call this the *Highest Uncertainty Method*.

4.3.3 RAD Slim

The RAD framework requires two models. Depending on the complexity of the models used, he cost of training might be excessive. Especially in scenarios relying on complex deep neural networks, such as Convolutional Neural Networks (CNNs) for image classification, it might be too expensive and time consuming to train two models. To reduce the computational cost we propose a slimmed version of RAD Active Learning named RAD Slim. The idea is to partially delegate the role of the label quality $\mathfrak{L}$ model to the oracle.

In RAD Slim new data batches arrive directly at the $\mathfrak{C}$ model, see Fig. 9. For each data instance we compare the given label k to the predicted label $\hat{k^{\mathfrak{C}}}$. If they are the same we add it to $\mathcal{D}^*$. If they differ we ask the oracle for the true label and add the answer to $\mathcal{S}^*$. To train the model $\mathfrak{C}_{i-1}$, we use only current $\mathcal{D}_i^*$ plus $\mathcal{S}_i^*$, not all the accumulated cleansed data as before. Considering computational cost, one pair of $\mathcal{D}_i^*$ and $\mathcal{S}_i^*$ will be used to train the model for 60 epochs. Optionally as before, we can impose a query limit, termed RAD Slim Limited. We query experts for the N_{lim} data instances with highest uncertainty ranked by decreasing cross-entropy loss between given label and prediction probability vector made by $\mathfrak{C}$. We call this the *Highest Loss Method*.

5 Experimental Evaluation

In this section, we implement RAD, RAD Voting and RAD Active Learning on IoT and Cluster datasets and report the

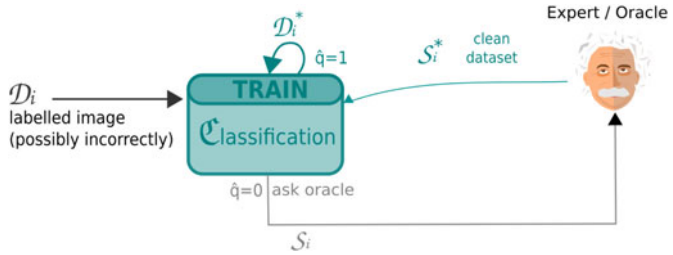


Fig. 9. RAD Slim.

evolution of learning accuracy under 30 and 40 percent noise level. For RAD, impact of noise level on final accuracy is discussed in Section 5.4. For RAD Voting, analysis on percentage of active and active-truth data changing over time is carried out in Section 5.7. RAD Active Learning and its small update RAD Active Learning Limited are explained in Section 5.8. The impact of the initial data batch size $|\mathcal{D}_0|$ on the above frameworks is studied in Section 5.9. To demonstrate the applicability of the framework to image dataset, RAD Slim and RAD Slim Limited are studied in Section 5.10.

5.1 Use Cases and Datasets

In order to demonstrate the general applicability of the proposed RAD framework for anomaly detection, we consider the following three use cases: (i) Cluster task failures, (ii) IoT botnet attacks and (iii) Face recognition. In our experiments, we use real data collected in cluster and IoT platforms and faces from real celebrity images.

The cluster task traces comprise data instances each corresponding to a task with 27 features capturing information related to static and dynamic system states, e.g., the task start/end times, the task resource utilisations, the hosting machine, etc. Each class is labeled based on its scheduling state. A detailed description of the features and labels can be found in [37]. In particular, we are interested in the four possible termination classes: *finish*, *fail*, *evict*, or *kill*. We filter out other classes. The resulting class distribution is dominated by successful tasks (*finish*) 77.8 percent, followed by *kill* 22.0 percent, *fail* 0.2 percent, and *evict* $< 0.1\%$. Similar to [39], we aim to predict the task outcome to reduce the resource waste and improve the overall scheduling and system performance, e.g., in case of lack of resources and the need to kill a task, help choosing the task with the least probability to succeed. We apply RAD to continually train a noise-resistant model for better accuracy. For this dataset we report the F1-score in addition to the accuracy due to the high class unbalance.

The IoT dataset comprises data instances describing 23 network packet-level statistics recursively computed over five different time scales totalling to 115 features. This traffic statistics are collected during normal operation, labeled as benign, or under one of ten different malicious attacks stemming from devices infected by either the *BASHLITE* or *Mirai* malware. All the classes are evenly distributed in the training and test dataset. Malicious traffic covers mainly scanning for vulnerable devices and various flooding attacks. The dataset provides traces collected at different IoT devices. More details are provided in [28]. We aim to apply RAD to build a noise-resistant model to categorize the attacks for post fact analysis, e.g., for threat assessment.

TABLE 2
Dataset Description

Use case	Cluster task failures	IoT device attacks	FaceScrub
#trainig data	60,000	33,000	12,000
#test data	6,000	6,000	3,000
#classes K	4	11	100
#features f	27	115	64*64
data batch size	600	300	2400
$\lvert\mathcal{D}_0\rvert$	6,000	6,000	2400

The FaceScrub [31] dataset is used for face recognition. Original FaceScrub contains more than 100,000 face images of 530 people, with about 200 images per person. Male and Female images are almost equal. We use a subset of 15K FaceScrub images to fit the limits of our compute resources. The 15K images cover the 100 people, 55 males and 45 females, with the highest number of images. On average each person (class) has 150 images with a standard deviation of 8.4 images. We use 12K images as training and 3K as test data. Training and test datasets have the same data distribution. FaceScrub images were retrieved from the Internet and are taken under real-world situations (uncontrolled conditions). We resize all images to 64x64 pixels. Name is the only annotation we use. Face recognition systems have been widely used in security equipment. We apply RAD Slimmed to FaceScrub dataset to show that our framework can also help to build robust face recognition models.

The main dataset characteristics are summarized in Table 2.

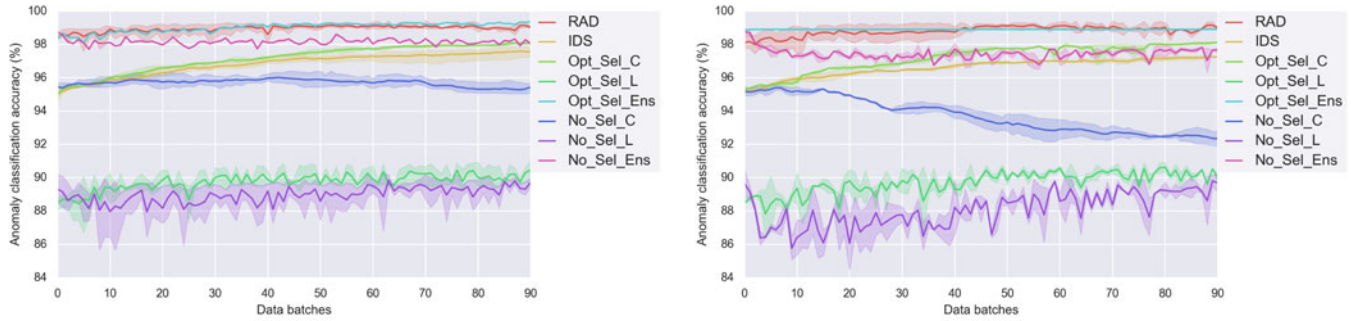
5.2 Experimental Setup

RAD is developed in Python using scikit-learn [33]. The main performance evaluation metric is accuracy. All results are averaged across three runs.

Noise. We inject noise into the two datasets by exchanging the true label of data instances with erroneous one. The label noise is symmetric, i.e., following the *noise completely at random* model (NCAR) in [13] where a label is picked with equal probability from all classes except the true one. The noise level $\tilde{Y}$ represents the percentage of data instances with noisy labels. We assume that all data is affected by label noise, except $\mathcal{D}_0$ and the testing data. We regenerate the noise at each experiment run.

Continual Learning. We start with an initial data batch of 6,000 data instances for the Cluster task failures and the IoT devices dataset. Then, data instances arrive continuously in batches of 600 (Cluster) and 300 (IoT) data instances. To kick-start the label and classification models in RAD we assume first batch contains only clean data, and subsequent data batches are affected by noise. We select 6,000 clean data instances as the test dataset for both use case. Test dataset will be used at the end of each epoch to evaluate the accuracy of the trained classification models. We show the evolution of the model accuracy over data batch arrivals until the performance of RAD converges.

Label Model. We use a multilayer perceptron to mainly assess the quality of each label, it will also be used to join the ensemble prediction. We fix the model initialization across experiment runs. For IoT and Cluster dataset, the



(a) With data noise level of 30%

(b) With data noise level of 40%

Fig. 10. Evolution of learning over time – Use case of IoT thermostat device attacks. Opt_Sel and No_Sel stand for optimal data selection and no filtering, respectively. _C, _L, and _Ens denote the model or strategy chosen for prediction.

neural network consists of two layers with 28 neurons each. The precision and robustness of the label model are critical to filter out the noisy labels and provide a clean training set to the classification model. We considered different models. Neural networks provided the best results in terms of accuracy and stability over time. Adaboost gave excellent accuracy when training from the initial data with ground truth, but it is too sensitive to label noise. Random forest is also known to be robust against label noise [13], however its accuracy was below the neural network one.

Classification Model. We use KNN to jointly do ensemble prediction with label model. For the extensions RAD Voting and RAD Active Learning, classification model will also play a role as label model to assess the quality of the data. We set the number of neighbours to five in KNN. Higher values can increase the resilience of the algorithm to residual noise, but also induce extra computational cost. The current choice stems from good results in preliminary experiments.

Slimmed Framework. For the face recognition task we use RAD Slim. In this case we use a 110-layers ResNet [18] as classification model. ResNet is a type of CNN architecture which introduces residual functions to alleviate the vanishing gradient problem in training deep neural networks improving the classification performance and model convergence. We use a fixed model initialization across experiment runs.

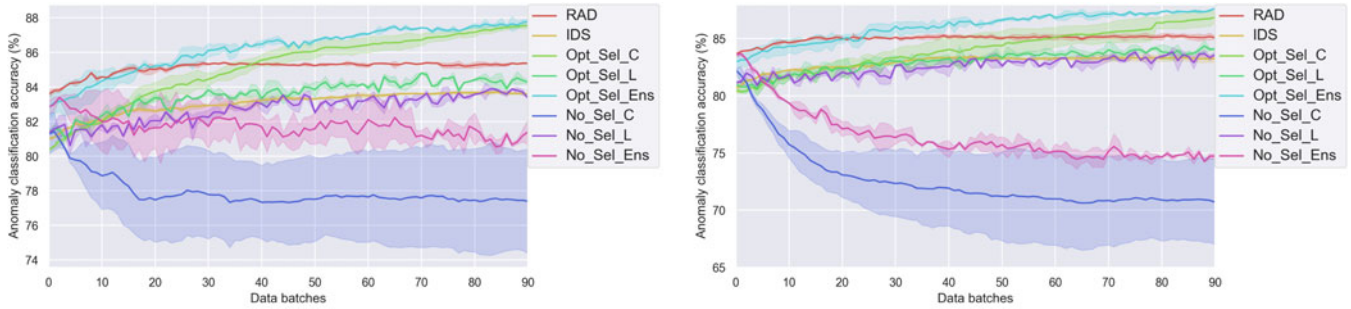
Baselines. The proposed RAD is compared against following baseline data selection schemes: 1) *No-Sel*, where all data instances of arriving batches are used for training the classification model; 2) *Opt-Sel* which emulates an omniscient agent who can perfectly distinguish between clean and noisy labels, and only use clean data to train the models; and, 3) *IDS*: the intrusion detection system from [1]. The main idea and structure of *IDS* are similar to the proposed RAD. The differences are: i) IDS only trains label quality model with $\mathcal{D}_0$ once without continuous updated; and, ii) IDS only uses classification model for predictions, instead of combining prediction results of quality and classification models. In addition, we consider: 4) *Full-Clean* which simulates perfectly recovered labels, i.e., all wrong labels have been correctly identified and recovered by, e.g., an oracle. This represents the ideal solution which provides all clean data in each data batch. In the following, model names ending in '_C' means the predictions are obtained from the anomaly classification model, ending in '_L' means the predictions are obtained from the label quality model, and ending in '_Ens' means the predictions are obtained from both anomaly classification and label quality model specified in Algorithm 1. *No-Sel*, *Opt-Sel* and *Full-Clean* use all the data to independently train the label quality and anomaly classification model. And Algorithm 1 is used to generate the final prediction. There is no filtering process in these cases.

To compare with RAD Slim on image dataset, we introduce two state-of-the-art approaches: 1) Forward [32] estimates the noise transition matrix before training the model, and subsequently uses this transition matrix for loss correction; and 2) Co-Teaching [17] trains two deep neural networks simultaneously to let them teach each other. For Forward, we use the same network architecture as for RAD Slim, i.e., 110-layers ResnNet. As Co-Teaching trains two models, we use two 56-layers ResNet. To speed up model convergence for RAD Slim, RAD Slim Limited, and Forward, we implement the E (Exponential)/PD (Proportional-Derivative)-Control [55] and Event-Based Control Learning rate [56] as learning rate schedule based on stochastic gradient descent (SGD) optimizer. Co-Teaching has its own learning rate scheduler.

5.3 Handling Dynamic Data

Figs. 10 and 11 show the evolution of the mean and variance of the classification accuracy achieved by RAD on the thermostat and task failure datasets, respectively. Each figure moreover presents results under two levels of label noise: 30 and 40 percent. We compare RAD against no selection (*No-Sel*), optimal selection (*Opt-Sel*) and *IDS*. One can notice that learning from all data instances without cleansing (i.e., *No-Sel* curves) gives consistently lower accuracy in all cases. For the task failure dataset, the accuracy even oscillates and diverges. The performance of RAD is better: (1) the accuracy does not diverge and (2) the accuracy consistently increases until it converges. The end accuracies, under 30 and 40 percent noise level, are all around 99 and 85 percent for the IoT attack and cluster tasks datasets, respectively. For the first dataset, the accuracy of RAD follows closely the ensemble prediction accuracy of *Opt-Sel*. As for the second dataset, RAD follows ensemble prediction of *Opt-Sel* at first but then converges after 30 data batch arrivals. Note that RAD gives also more stable results as shown by shorter variance bars which in magnitude are in line with the ones obtained by an ideal data cleansing. For *No-Sel* the bars are significantly larger.

We note that ensemble prediction can greatly enhance the learning outcomes in the presence of noisy data, compared



(a) With data noise level of 30% (b) With data noise level of 40%

Fig. 11. Evolution of learning over time – Use case of Cluster task failures.

the prediction of solely the label quality or classification model. Such an observation holds for the different data selection schemes discussed in the subsequent sections. Due to space limits, we skip the presentation of those results.

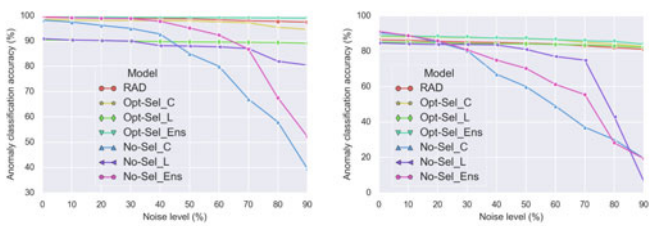
In summary: (i) continual learning is advantageous compared to using only the initial dataset; however, (ii) continual learning exposes to possible classification accuracy degradation stemming from noisy labels if proper data selection is lacking, (iii) RAD improves the classification accuracy compared to taking all labels, (iv) the data selection of RAD is good, and close to being optimal in some cases, and (v) ensemble prediction can greatly enhance the robustness against noisy data.

5.4 Evaluation of Noise Robustness of RAD

Next we investigate the impact of different noise levels on the RAD performance in terms of classification accuracy.

Figs. 12a and 12b present the classification accuracy for various levels of noise, ranging from 0 percent (all data are clean) up to 90 percent for our two main reference datasets: IoT thermostat device attacks and Cluster task failures. All experiment settings remain the same as before, only the noise level of training data batches varies. Once again, the RAD performance is compared to learning from all data (*No-Sel*) and an omniscient data cleanser (*Opt-Sel*).

As illustrated in Section 3, for *No-Sel* the noisier the data are, the worse the classification accuracy, with ensemble prediction, dropping to 20 and 52 percent for the Cluster and IoT datasets, respectively. A decreasing trend can also be found for RAD and *Opt-Sel*, however the drops are significantly smaller: at most 5 percent. As there is by definition no noise in *Opt-Sel* case, the decrease in classification accuracy is only due to the reduction of the overall amount of clean data to learn from. Since the data cleansing of RAD is not perfect, the accuracy reduction is caused by noise



(a) IoT thermostat device attacks (b) Cluster task failures

Fig. 12. Impact of data noises on RAD accuracy.

pollution and overall clean data reduction. Nevertheless, the impact is small and any huge accuracy pitfall is avoided which results in RAD's performance being close to *Opt-Sel*. We can conclude that RAD can limit the impact of the amount of noise across a wide range of noise levels.

5.5 Analysis of All Datasets

Summary results are reported in Table 3. One can see that results of RAD are always better than IDS and *No-Sel*. For IoT dataset with 40 percent noise, RAD is even better than any single model of *Opt-Sel*. But there is still room for improvement between RAD and *Opt-Sel_Ens*. F1 scores are consistent with accuracy results with few exceptions. Under the cluster dataset and 30 percent noise, F1 score of *No-Sel_Ens* is better than *No-Sel_L* and *No-Sel_C* even if the accuracy is slightly worse than *No-Sel_L*. Under 40 percent noise, the accuracy of *No-Sel_Ens* is 8.43 points worse than *No-Sel_L*, however the difference in F1 score is only 0.01. This means that the accuracy difference is mostly due to data unbalance.

The resilience to high levels of noise might be even more important than the benefits of continual learning. Under such levels, the classification accuracy without data cleansing diverges for all datasets. Even if it is rare to have noise levels of 90 percent or above, they might still happen for short time periods in case of attacks to the auto-labelling system, e.g., via flooding of malicious labels. Hence this property can be crucial for the dependability of the auto-labelling system.

5.6 Limitation of RAD Framework

Though RAD works well for datasets of Cluster task failures and IoT device attacks. We can still see the potential limitations of this framework. For example: 1) the assumption of availability of a small fraction of clean data which may not be possible; 2) if data is coming at high rates, training two models simultaneously instead of one can slow down the system; 3) as the anomaly classifier receives only the data selected by the label model, there is a risk that the classifier model overfits to label model. To address these issues we devised the two extensions presented in Section 4.3. These are evaluated in the next subsections.

5.7 RAD Voting and History Extension

In the first extension we let both the label and classifier models vote on the label quality and include the possibility to recover instances from history to be evaluated as the model performances improve over time.

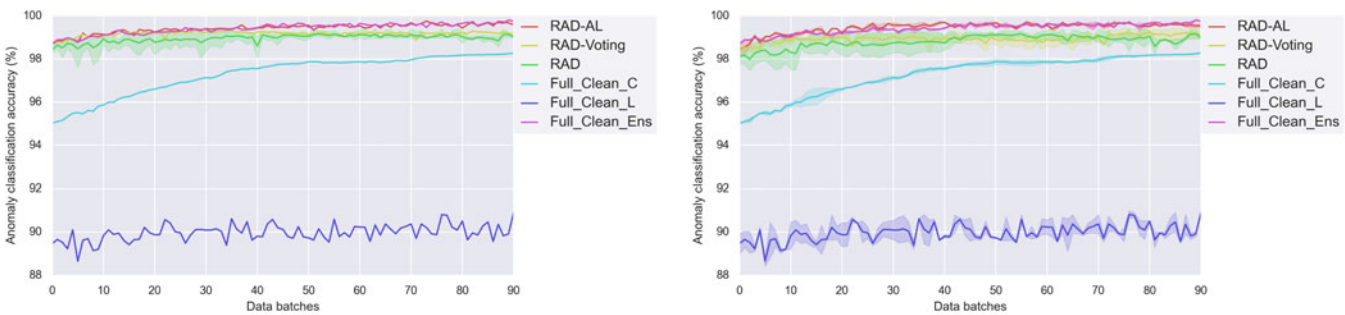
TABLE 3
Final Accuracy of All Algorithms for the Cluster Task Failures and IoT Device Attacks Datasets on 30 and 40 Percent Noise Level

Algorithm	Cluster (30%)	IoT(30%)	Cluster (40%)	IoT(40%)
Full-Clean_C	89.35(0.90)	98.28	89.35(0.90)	98.28
Full-Clean_L	85.17(0.84)	90.87	85.17(0.84)	90.87
Full-Clean_Ens	91.08(0.91)	99.83	91.08(0.91)	99.83
Opt-Sel_C	87.68(0.87)	98.08	87.16(0.87)	98.06
Opt-Sel_L	84.37(0.82)	90.81	84.18(0.83)	89.70
Opt-Sel_Ens	87.88(0.87)	99.35	87.60(0.87)	99.25
No-Sel_C	77.40(0.79)	95.47	71.02(0.74)	92.27
No-Sel_L	83.54(0.82)	89.95	83.35(0.80)	89.57
No-Sel_Ens	81.53(0.83)	98.06	74.92(0.79)	97.51
RAD	85.46(0.84)	99.01	85.03(0.83)	98.95
IDS	83.63(0.81)	97.83	83.31(0.81)	97.23
RAD Voting	86.01(0.85)	99.21	85.73(0.84)	99.07
RAD-AL[1]	90.84(0.90)	99.72	90.77(0.90)	99.58
RAD-AL-L[2]	90.00(0.90)	99.68	-	-
PSO[3]	87.83(0.87)	98.85	-	-

1. RAD-AL: RAD Active Learning.
2. RAD-AL-L: RAD Active Learning Limited.
3. PSO: Pre-Select Oracle.
Final F1-score is reported in brackets for the Cluster dataset. All results are averaged across 3 runs.

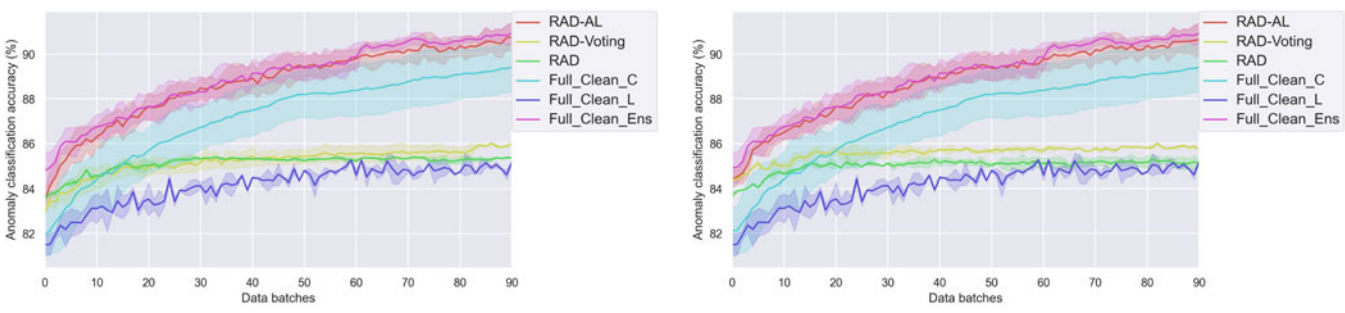
We evaluate the accuracy of RAD Voting over time and different noise levels in Figs. 13 and 14 for the IoT thermostat and Cluster task failures, respectively. For the IoT dataset, RAD Voting is better than any single model of *Full-Clean*. For the Cluster dataset, RAD Voting does not converge as RAD. Table 3 summarizes and compares the RAD Voting performance with others. We can see that RAD Voting performance is always better than RAD. This is because we correct labels in the RAD Voting algorithm, which increases the number of training instances over RAD. The F1-score results of the Cluster dataset are in line with the accuracy results.

To better understand the different performance between the two datasets we define A (called Hot) as the percent of data used for training till time t_i



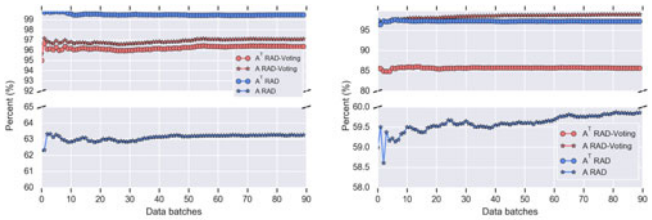
(a) Iot data with noise level of 30% (b) Iot data with noise level of 40%

Fig. 13. Evolution of learning over time – Use case of IoT thermostat device attacks with RAD Voting and RAD Active Learning (RAD-AL). Full_clean means that no label noise is injected.



(a) Cluster data with noise level of 30% (b) Cluster data with noise level of 40%

Fig. 14. Evolution of learning over time – Use case of Cluster task failures with RAD Voting and RAD Active Learning.



(a) IoT data with 30% noise (b) Cluster data with 30% noise

Fig. 15. RAD and RAD Voting: percentage of hot data and how-truth it is.

$$A = \frac{\sum_{k=1}^{i}(|\mathcal{D}_k^*| + |\mathcal{U}_k^*|)}{\sum_{k=1}^{i}|\mathcal{D}_k|}. \quad (1)$$

Knowing the number of true clean labels used per batch C_i^T, we further define A^T (called Hot-Truth) as the percent of true clean active data

$$A^T = \frac{\sum_{k=1}^{i} C_k^T}{\sum_{k=1}^{i}(|\mathcal{D}_k^*| + |\mathcal{U}_k^*|)}. \quad (2)$$

In both formulas, we exclude the initial clean batch $\mathcal{D}_0$. Intuitively, A tells how much of the incoming data we use for training, and A^T how clean the used training data is.

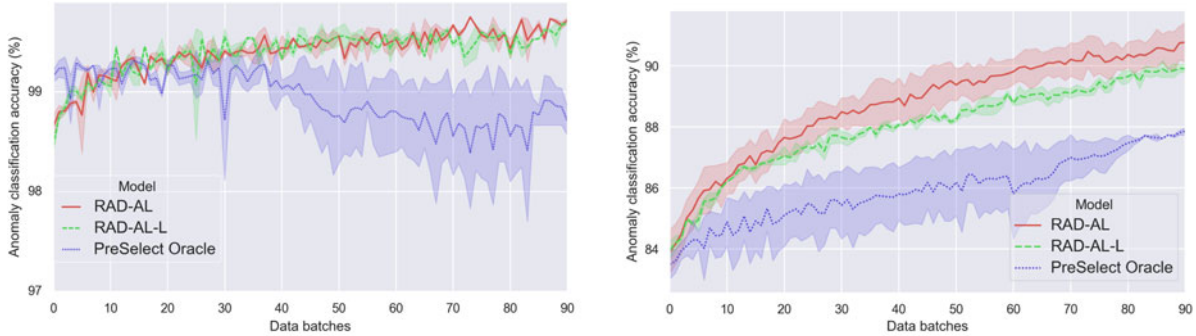
Figs. 15a and 15b plot A and A^T using RAD and RAD Voting over time for the IoT and Cluster datasets, respectively. For the IoT dataset both A and A^T improve over time with RAD Voting, see Fig. 15a. This means that both the quantity of active data, i.e., A, and the quality, i.e., cleanliness, of the active data A^T improve over time. For Cluster dataset, A^T of RAD Voting does not improve over time even though A increases, see Fig. 15b. We attribute this to the fact that both $\mathfrak{C}$ and $\mathcal{L}$ predict the same wrong class and this class is used to replace the original label of the data instance. Next we compare the performance of RAD and RAD Voting. One can see that RAD Voting includes more data into the training set (higher A) than RAD, but the data is less clean (lower A^T). Overall since RAD Voting filters out less training data (final A of RAD Voting is 27.74 and 39.07 percent higher than RAD for the IoT and Cluster experiments, respectively), the difference in A^T is relatively small (final A^T of RAD Voting is only 3.11 and 11.56 percent smaller than RAD for the IoT and Cluster experiments, respectively). $A^T \cdot A$ reflects the ratio between clean label data used in training and the total received data. Both RAD and RAD-voting receive the same amount of data in every epoch. Therefore the higher $A^T \cdot A$ is, the more clean label data is used in training. For both use cases, the final $A^T \cdot A$ of RAD Voting is higher than RAD. Intuitively RAD Voting should be better than RAD and experiment results are in line with this intuition.

5.8 RAD Active Learning

RAD Active Learning extends RAD with the ability of asking an oracle to provide the true label for data instances where the two models disagree. First we consider RAD Active Learning with no limits on the number of oracle requests followed by RAD Active Learning Limited which limits the number of oracle interactions.

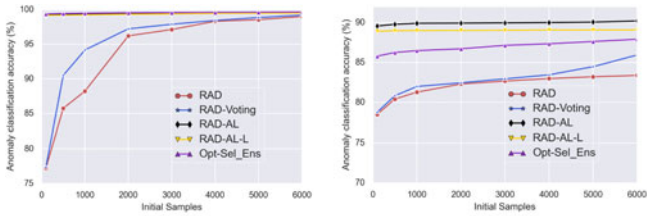
Figs. 13 and 14 show the performance of RAD Active Learning (RAD-AL) for the IoT and Cluster datasets under 30 and 40 percent noise, respectively. The figures compare RAD Active Learning to RAD Voting, *No-Sel* and *Full-Clean*. We can see that RAD Active Learning is always better than RAD Voting and almost as good as *Full-Clean_Ens* across the two different datasets and different noise levels. From the results in Table 3, we can observe that the result of RAD Active Learning is extremely close to *Full-Clean_Ens* who is the best in every column. That shows that our training data selection in RAD Active Learning is very accurate. Almost all noisy data are filtered out for consultations to expert.

Consulting every single uncertain data instance with expert might be too expensive or impossible in practice. Hence, we consider RAD Active Learning Limited which limits the consultations with experts. Here we limit the number of queries per batch to 20 percent of the batch size. To illustrate the power of our training data selection process, we introduce a new comparison: Pre-Select Oracle. Pre-Select Oracle has the same number of consultation to oracles as RAD Active Learning Limited, but data instances are selected randomly before training. Fig. 16 shows the results for IoT and Cluster datasets, one can notice that the curve of RAD Active Learning Limited (RAD-AL-L) increases along with RAD Active Learning and largely outperforms Pre-Select Oracle. The accuracy difference here is due to the uncertainty ranking used by the Highest Uncertainty Method. From the result in Table 3, one can see that after imposing a query limit of 20 percent, RAD Active Learning Limited reaches similar accuracy as RAD Active Learning, and higher accuracy than RAD and RAD Voting.



(a) Iot data with noise level of 30% (b) Cluster data with noise level of 30%

Fig. 16. Comparison of RAD Active Learning Limited (RAD-AL-L) and pre-select Oracle, showing the power of selection.



(a) IoT thermostat device attacks (b) Cluster task failures

Fig. 17. Impact of size of initial data batch $\mathcal{D}_0$ on RAD accuracy with 30 percent noise level.

5.9 Impact of Initialization

Here we study the impact on RAD and its extensions of the size of the initial dataset $\mathcal{D}_0$. We vary the number of initial clean data instances from 100 to 6,000, and measure the classification accuracy after 90 data batch arrivals. We consider the *Opt-Sel* baseline since the *No-Sel* baseline is meant for the framework configuration, not its performance evaluation.

Figs. 17a and 17b show the results for the IoT and Cluster datasets, respectively. *Opt-Sel_Ens* seems to perform independent from the number of initial data instances ($|\mathcal{D}_0|$) in Fig. 17a. This is due to the fact that after 90 batch arrivals the amount of training data is sufficient for the accuracy to converge. In Fig. 17b however, we can see that $|D_0|$ influences the accuracy of *Opt-Sel_Ens*. The model is yet to converge at the end of learning, but the influence is clearly smaller than that for RAD and RAD Voting. For these two models the size of $\mathcal{D}_0$ matters more: the larger the better. At $|\mathcal{D}_0| = 2000$ their performances are similar to *Opt-Sel_Ens* (less than 5 percent difference) for IoT dataset, and at $|\mathcal{D}_0| = 6000$ they almost overlap. RAD Voting outperforms RAD in both datasets under all sizes of $\mathcal{D}_0$. This is because RAD Voting can correct data labels and thus increase the number of training instances. Finally, RAD Active Learning and RAD Active Learning Limited (20 percent limit) do not depend on the size of $\mathcal{D}_0$, since they can ask the oracle for the label of uncertain data instances.

This justifies our earlier choice of $\mathcal{D}_0$ having 6,000 data instances as it enables to achieve the best accuracy. However, all proposed frameworks could also perform well with only half the initial data instances in $\mathcal{D}_0$.

5.10 RAD Slim on Image Data

We evaluate the RAD framework on the challenging case of noisy image classification. Specifically, we apply RAD Slim and RAD Slim Limited (20 percent of batch size query limit per batch) to train a classifier that encounters on-line noisy images. Fig. 18 shows the accuracy results across the batch arrivals. We can observe that RAD Slim is close to the *Full-Clean* baseline and largely outperforms other baselines. Detailed numbers are summarized in Table 4. Fig. 19 shows the comparison between RAD Slim, RAD Slim Limited and Pre-Select Oracle (same design as in Section 5.8). One can see that RAD Slim Limited performs significantly better than Pre-Select Oracle when both use the same limit on the number of expert queries.

To further display the effectiveness of RAD Slim on different types of attack, we design a series of unbalanced noisy data batches. The original Facescrub dataset comprises a ratio of 55%:45% male and female images. For $\mathcal{D}_0$ of

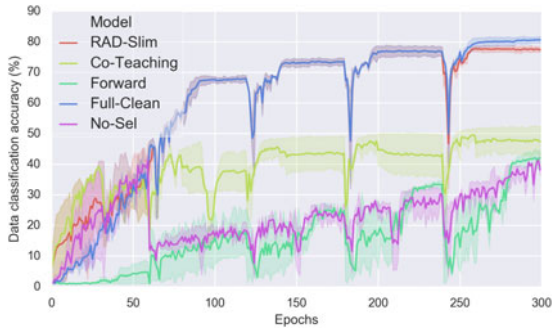


Fig. 18. FaceScrub with noise level of 30 percent.

TABLE 4
Final Accuracy of Different Algorithms on FaceScrub Dataset With 30 percent Noise, Results are Averaged on 3 Runs

Algorithm	Accuracy
Full-Clean	81.72
No-Sel	38.89
Forward	41.71
Co-Teaching	47.39
RAD Slim	77.51
RAD Slim Limited	67.18
Pre-Select Oracle	52.12
RAD Slim-Unbalanced	66.12
Unbalanced-No-Sel	37.11
Unbalanced-Full-Clean	69.95

the unbalanced data batches, image ratio of male and female is 90%:10% followed by 45%:55% in subsequent batches. Fig. 20 shows the results, RAD Slim performs definitely better than no selection, and very close to full clean scenario, which shows that RAD Active Learning can not only defend the model from different noise levels, but also resist other types of attack.

Another observation is that all curves suffer a periodic up-down pattern. This is because for image dataset, each time a new batch comes, we only use this new batch data as training dataset. As different batches provide different subviews of the data the empirical distribution can be different from the calculated optimum, but the model remains. So for the first epoch of a new batch, we will generate a gradient

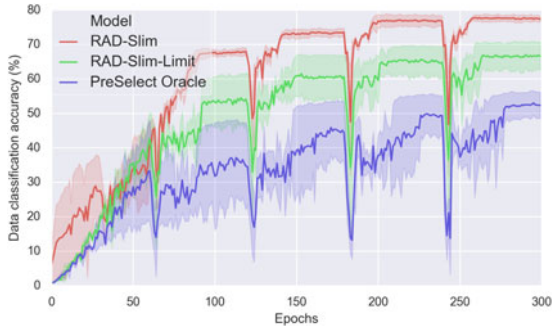


Fig. 19. RAD Slim Limited on FaceScrub with 30 percent noise.

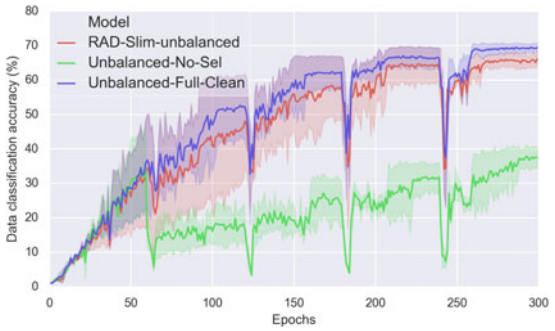


Fig. 20. Unbalanced FaceScrub with 30 percent noise.

which is based on new data but applied on an old model. This can influence the accuracy of the model. Moreover when retraining on each new data batch we reset the learning rate which causes a bump in the learning rate. Therefore, even if all batches follow the same distribution, the system could temporarily wander off from the previous optimum.

6 Concluding Remarks

While machine learning classification algorithms are widely applied to detect anomalies, the commonly employed assumption of clean anomaly labels often does not hold for data collected in the wild due to careless annotation and malicious dirty label pollution. The noisy labels can significantly degrade the accuracy of anomaly detection and are challenging to tackle due to the lack of ground truth of label quality. In this paper, we present a on-line framework for robust anomaly detection, RAD, which can continuously learn the system dynamics and anomaly behaviours from streams of arriving data after filtering out suspicious noisy data.

RAD is a general framework that composes of label quality predictor and classification model, where the former mainly captures the label dynamics and the latter focuses on increasing the diversity of prediction. Predictions from both contribute to the final decision on detecting anomaly. To adapt to the on-line nature of anomaly detection, we extend RAD with additional features of conflicting opinions of classifiers, repetitively cleaning, and oracle knowledge, corresponding to RAD Voting, RAD Active Learning, and RAD Active Learning Limited. We demonstrate the effectiveness of RAD and its extensions on three uses cases, i.e., detecting IoT device attacks, predicting task failures at Google clusters and recognising celebrity faces from FaceScrub. The evaluation results on three use cases show remarkable accuracy that are close to the case without encountering anomaly input. In short, RAD is a general robust learning framework that can be applied on different classification models and enhances their robustness against noisy inputs during on-line training.

Acknowledgments

This work was supported in part by the French LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01), the Swiss National Science Foundation NRP75 Project 407540_167266 and the National Natural Science Foundation of China (Grant No. 61872337).

References

[1] M. Agarwal, D. Pasumarthi, S. Biswas, and S. Nandi, "Machine learning approach for detection of flooding DoS attacks in 802.11 networks and attacker localization," *Int. J. Mach. Learn. Cybern.*, vol. 7, no. 6, pp. 1035–1051, 2016.

[2] W. An and M. Liang, "Fuzzy support vector machine based on within-class scatter for classification problems with outliers or noises," *Neurocomputing*, vol. 110, pp. 101–110, Jun. 2013.

[3] M. Anbar, R. Abdullah, B. N. Al-Tamimi, and A. Hussain, "A machine learning approach to detect router advertisement flooding attacks in next-generation IPv6 networks," *Cogn. Comput.*, vol. 10, no. 2, pp. 201–214, 2018.

[4] S. Banescu, C. S. Collberg, and A. Pretschner, "Predicting the resilience of obfuscated code against symbolic execution attacks via machine learning," in *Proc. 26th USENIX Secur. Symp.*, 2017, pp. 661–678.

[5] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Proc. Asian Conf. Mach. Learn.*, 2011, pp. 97–112.

[6] R. Birke, G. Ioana, L. Y. Chen, D. Wiesmann, and T. Engbersen, "Failure analysis of virtual and physical machines: Patterns, causes and characteristics," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2014, pp. 1–12.

[7] R. Birke, A. Podzimek, L. Y. Chen, and E. Smirni, "Virtualization in the private cloud: State of the practice," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 608–621, Sep. 2016.

[8] C. E. Brodley and M. A. Friedl, "Improving automated land cover mapping by identifying and eliminating mislabeled observations from training data," in *Proc. Int. Geosci. Remote Sens. Symp.*, 1996, pp. 1379–1381.

[9] J. R. Campos, M. Vieira, and E. Costa, "Exploratory study of machine learning techniques for supporting failure prediction," in *Proc. 14th Eur. Dependable Comput. Conf.*, 2018, pp. 9–16.

[10] S. Cerf, R. Birke, and L. Y. Chen, "Duo learning for classifications with noisy labels," in *Proc. Adv. Conf. Neural Inf. Process. Syst.*, 2018.

[11] Y. Fan, J. Li, and D. Zhang, "A method for identifying critical elements of a cyber-physical system under data attack," *IEEE Access*, vol. 6, pp. 16972–16984, 2018.

[12] Z. Fang, J. Tzeng, C. C. Chen, and T. Chou, "A study of machine learning models in epidemic surveillance: Using the query logs of search engines," in *Proc. Pacific Asia Conf. Inf. Syst.*, 2010, Art. no. 137.

[13] B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 845–869, May 2014.

[14] A. Ghiassi, T. Younesian, Z. Zhao, R. Birke, V. Schiavoni, and L. Y. Chen, "Robust (deep) learning framework against dirty labels and beyond," in *Proc. IEEE Int. Conf. Trust Privacy Secur. Intell. Syst. Appl.*, 2019, pp. 236–244.

[15] G. Giantamidis and S. Tripakis, "Learning moore machines from input-output traces," in *Proc. 21st Int. Symp. Formal Methods*, 2016, pp. 291–309.

[16] I. Guyon, N. Matić, and V. Vapnik, "Discovering informative patterns and data cleaning," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 1994, pp. 145–156.

[17] B. Han *et al.*, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 8536–8546.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[19] Y. He, G. J. Mendis, and J. Wei, "Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2505–2516, Sep. 2017.

[20] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel, "Using trusted data to train deep networks on labels corrupted by severe noise," in *Proc. Conf. Neural Inf. Process. Syst.*, 2018, pp. 10477–10486.

[21] T. H. Huang, C. Yu, and H. Kao, "Data-driven and deep learning methodology for deceptive advertising and phone scams detection," in *Proc. Conf. Technol. Appl. Artif. Intell.*, 2017, pp. 166–171.

[22] P. Jeatrakul, K. Wong, and C. Fung, "Data cleaning for classification using misclassification analysis," *J. Adv. Comput. Intell. Intell. Inform.*, vol. 14, pp. 297–302, Apr. 2010.

[23] J.-W. Kang, I.-Y. Joo, and D.-H. Choi, "False data injection attacks on contingency analysis: Attack strategies and impact assessment," *IEEE Access*, vol. 6, pp. 8841–8851, 2018.
[24] D. Karagiannis and A. Argyriou, "Jamming attack detection in a pair of RF communicating vehicles using unsupervised machine learning," *Veh. Commun.*, vol. 13, pp. 56–63, 2018.
[25] R. Kozik, M. Choras, M. Ficco, and F. Palmieri, "A scalable distributed machine learning approach for attack detection in edge computing environments," *J. Parallel Distrib. Comput.*, vol. 119, pp. 18–26, 2018.
[26] J. Larsen, L. Nonboe, M. Hintz-Madsen, and L. K. Hansen, "Design of robust neural network classifiers," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 1998, pp. 1205–1208.
[27] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and L.-J. Li, "Learning from noisy labels with distillation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1928–1936.
[28] Y. Meidan *et al.* "N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Third Quarter 2018.
[29] A. L. B. Miranda, L. P. F. Garcia, A. C. P. L. F. de Carvalho, and A. C. Lorena, "Use of classification algorithms in noise detection and elimination," in *Proc. 4th Int. Conf. Hybrid Artif. Intell. Syst.*, 2009, pp. 417–424.
[30] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 1196–1204.
[31] H.-W. Ng and S. Winkler, "A data-driven approach to cleaning large face datasets," in *Proc. IEEE Int. Conf. Image Process.*, 2015, pp. 343–347.
[32] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2233–2241.
[33] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[34] A. Pellegrini, P. di Sanzo, and D. R. Avresky, "A machine learning-based framework for building application failure prediction models," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshop*, 2015, pp. 1072–1081.
[35] C. Pham *et al.*, "Failure diagnosis for distributed systems using targeted fault injection," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 2, pp. 503–516, Feb. 2017.
[36] T. Pitakrat, A. van Hoorn, and L. Grunske, "A comparison of machine learning algorithms for proactive hard disk drive failure detection," in *Proc. 4th Int. ACM SIGSOFT Symp. Architecting Crit. Syst.*, 2013, pp. 1–10.
[37] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: Format+ schema," Google Inc., White Paper, pp. 1–14, 2011.
[38] U. Reuter, A. Sultan, and D. S. Reischl, "A comparative study of machine learning approaches for modeling concrete failure surfaces," *Advances Eng. Softw.*, vol. 116, pp. 67–79, 2018.
[39] A. Rosà, L. Y. Chen, and W. Binder, "Failure analysis and prediction for big-data systems," *IEEE Trans. Services Comput.*, vol. 10, no. 6, pp. 984–998, Nov./Dec. 2017.
[40] A. Rosà, L. Y. Chen, and W. Binder, "Understanding the dark side of big data clusters: An analysis beyond failures," in *Proc. 45th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2015, pp. 207–218.
[41] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 815–823.
[42] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015.
[43] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Training convolutional networks with noisy labels," 2014, *arXiv:1406.2080*.
[44] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1701–1708.
[45] J. Thongkam, G. Xu, Y. Zhang, and F. Huang, "Support vector machine for outlier detection in breast cancer survivability prediction," in *Proc. Int. Workshops Adv. Web Netw. Technol. Appl.*, 2008, pp. 99–109.
[46] V. N. Vagin and M. V. Fomina, "Problem of knowledge discovery in noisy databases," *Int. J. Mach. Learn. Cybern.*, vol. 2, no. 3, pp. 135–145, 2011.
[47] A. Vahdat, "Toward robustness against label noise in training deep discriminative neural networks," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5601–5610.
[48] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie, "Learning from noisy large-scale datasets with minimal supervision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6575–6583.
[49] H. Wang *et al.*, "CosFace: Large margin cosine loss for deep face recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5265–5274.
[50] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-basedlearning algorithms," *Mach. Learn.*, vol. 38, no. 3, pp. 257–286, Mar. 2000.
[51] J. Xue, R. Birke, L. Y. Chen, and E. Smirni, "Spatial-temporal prediction models for active ticket managing in data centers," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 1, pp. 39–52, Mar. 2018.
[52] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.
[53] M. Zhang, T. Li, H. Shi, Y. Li, and P. Hui, "A decomposition approach for urban anomaly detection across spatiotemporal data," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 6043–6049.
[54] Z. Zhao *et al.* "Robust anomaly detection on unreliable data," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2019, pp. 630–637.
[55] Z. Zhao, S. Cerf, B. Robu, and N. Marchand, "Feedback control for online training of neural networks," in *Proc. IEEE Conf. Control Technol. Appl.*, 2019, pp. 136–141.
[56] Z. Zhao, S. Cerf, B. Robu, and N. Marchand, "Event-based control for online training of neural networks," *IEEE Control Syst. Lett.*, vol. 4, no. 3, pp. 773–778, Jul. 2020.
[57] B. Zhou, J. Li, J. Wu, S. Guo, Y. Gu, and Z. Li, "Machine-learning-based online distributed denial-of-service attack detection using spark streaming," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–6.

Zilong Zhao received the PhD degree in automation and information from the Université Grenoble Alpes (UGA), Grenoble, France. He is a postdoctoral researcher with the Delft University of Technology, The Netherlands since 2020. He was the lead AI and data engineer at the Miro Health. His research focuses on applying control theory into machine learning algorithms, building robust and dependable machine learning systems, AI application optimization, and generative adversarial networks (GANs).

Robert Birke (Senior Member, IEEE) received the PhD degree in electronics and communications engineering from the Politecnico di Torino, Turin, Italy. He is a principal scientist with the ABB Research Lab, Switzerland. His research interests include the broad area of virtual resource management including network design, workload characterization, and AI and big-data application optimization. He has published more than 80 papers at venues related to communication and system performance, e.g., SIGCOMM, SIGMETRICS, FAST, INFOCOM, and the *IEEE Journal on Selected Areas in Communications*.

Rui Han received the MSc (honor) degree from Tsinghua University, Beijing, China, in 2010, and the PhD degree from the Department of Computing, Imperial College London, London, U.K., in 2014. He is an associate professor with the School of Computer Science and Technology, Beijing Institute of Technology, China. His research interests include system optimization for deep learning workloads. He has more than 40 publications in these areas, including papers at the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Computers*, INFOCOM, and ICDCS.

Bogdan Robu received the PhD degree from the University of Toulouse, Toulouse, France, in 2010. He is an associate professor at the Université Grenoble Alpes (UGA) and a researcher with GIPSA-Lab Laboratory, Grenoble, France since September 2011. His research focus is on applying control theory techniques to machine learning algorithms, Cloud/Fog software and parallel computing systems in order to achieve dependable, trustworthy and highly available systems. He has co-authored around 40 publications in peer-reviewed conferences and journals.

Sara Bouchenak is professor with INSA Lyon – LIRIS Laboratory. She conducts research on highly available, dependable, privacy preserving, and efficient distributed computer systems. She serves as Computer Science Committee chair for INSA Lyon – Department of Computer Science. She has co-authored more than 70 publications. She has been chair and member of the PC of several conferences (ATC, DSN, ICDCS, SRDS, The WebConf, etc.). She has coordinated and participated in several national and international projects. She serves as scientific expert for the European Commission, ANR (France), FNS (Switzerland), and Vinnova (Sweden).

Sonia Ben Mokhtar received the PhD degree from the Université Pierre et Marie Curie, Paris, France, in 2007 before spending two years at the University College London, London, U.K. She is a CNRS research director with the LIRIS Lab and the head of the Distributed Systems and Information Retrieval Group (DRIM). Her research focuses on the design of resilient and privacy-preserving distributed systems. She has co-authored more than 70 papers in peer-reviewed conferences and journals and has served on the editorial board of the *IEEE Transactions on Dependable and Secure Computing*.

Lydia Y. Chen (Senior Member, IEEE) received the BA degree from the National Taiwan University, Taipei, Taiwan, and the PhD degree from Pennsylvania State University, State College, Pennsylvania. She is an associate professor with the Department of Computer Science, Delft University of Technology, The Netherlands. Prior to joining TU Delft, she was a research staff member with the IBM Research Zurich Lab from 2007 to 2018. Her research interests include distributed machine learning, dependability management, resource allocation for large-scale data processing systems and services. More specifically, her work focuses on developing stochastic and machine learning models, and applying these techniques to application domains, such as data centers and AI systems. She has published more than 100 papers in peer-reviewed journals. She has served on the editorial boards of the *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Service Computing*, and *IEEE Transactions on Network and Service Management*.