Incipient anomalies present milder symptoms compared to severe ones, and are more difficult to detect and diagnose due to their close resemblance to normal operating conditions. The lack of incipient anomaly examples in the training data can pose severe risks to anomaly detection methods that are built upon Machine Learning (ML) techniques, because these anomalies can be easily mistaken as normal operating conditions. To address this challenge, we propose to utilize the uncertainty information available from ensemble learning to identify potential misclassified incipient anomalies. We show in this paper that ensemble learning methods can give improved performance on incipient anomalies and identify common pitfalls in these models through extensive experiments on two real-world datasets. Then, we discuss how to design more effective ensemble models for detecting incipient anomalies.

CCS Concepts: • Computing methodologies → Ensemble methods; Computer vision; Supervised learning by classification; • Computer systems organization → Reliability; Embedded and cyber-physical systems.

Additional Key Words and Phrases: Incipient anomalies, Fault Detection and Diagnosis (FDD), uncertainty estimation

ACM Reference Format:
Baihong Jin, Yingshui Tan, Albert Liu, Xiangyu Yue, Yuxin Chen, and Alberto Sangiovanni Vincentelli. 2020. Using Ensemble Classifiers to Detect Incipient Anomalies. 1, 1 (August 2020), 25 pages. https://doi.org/10.1145/nnnnnnnnnnnnnnn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2020 Association for Computing Machinery.
XXXX-XXXX/2020/8-ART $15.00
https://doi.org/10.1145/nnnnnnnnnnnnnnn
Fig. 1. (a) Visualization of part of the RP-1043 chiller data [5] where the “severity spectrum” for two fault conditions is visible. The normal condition and two fault conditions (each with four SLs) are shown. (b) Illustration showing how an ensemble classifier can conceptually help detect incipient anomalies. The gray lines represent the decision boundaries of base learners in the ensemble.

1 INTRODUCTION

In anomaly detection applications\(^1\), it is common to encounter anomaly data examples whose symptoms correspond to different Severity Levels (SLs). Fig. 1a shows a real-world example where faults are categorized into four different SLs, from SL1 (slightest) to SL4 (most severe). The ability of accurately assessing the severity of faults/diseases is important for anomaly detection applications, yet very difficult on low-severity examples; SL1 data clusters are much closer to the normal cluster than to their corresponding SL4 clusters in Fig. 1a. A anomaly detection system needs to be very sensitive to identify the low-severity faults; at the same time, it should keep the number of false positives low, which makes the design of such decision systems a challenging task.

If labeled data from different SLs are available, then regular regression or classification approaches are suitable, as already exemplified by previous research [19, 24]. However, these fine-grained labeled datasets can take much effort to prepare and we may not always have \textit{a priori} access to a full spectrum of anomaly SLs. In an extreme case, as illustrated in Fig. 1b, suppose we only have access to the two ends (i.e. the normal condition SL0 and the most severe anomaly condition SL4) of the severity spectrum; incipient anomaly instances are not available to us. If we train a classification system only using the available SL0 and SL4 data, the resulting classifier may have great performance on in-distribution data (SL0 & SL4). However, it may fail badly with identifying the incipient anomaly data. For example, most SL1 faults may be mistakenly recognized as normal by any of the decision boundaries shown in Fig. 1b. More generally, classical supervised learning approaches designed for achieving maximal separation between labeled classes (e.g. margin-based classifiers, discriminative neural networks, etc), are less effective in detecting such low-severity, incipient anomaly data examples.

In the absence of labeled data for certain categories of fault instances, common practices are to develop generative models, such as the Gaussian mixture model [37], Principle Component Analysis (PCA) [15, 35], Long Short-Term Memory (LSTM) network [8] and autoencoder [32, 34].

\(^1\)In this paper, an “anomaly” can mean either a machine fault in industrial applications or a human disease in health applications.
A potential problem for these models is that they may not always generalize well—that is, a single-trained model, when applied to an unseen incipient anomaly instance at test time, can be classified as normal [7], i.e. becoming a false negative.

The solution we propose in this paper is based on ensemble learning [36], i.e., on the process of training multiple classifiers and leveraging their joint decisions to recognize incipient anomalies. In literature, a variety of ensemble methods have been proposed on the estimation of decision uncertainties [9, 20, 21]. Fig. 1b shows that the individual classifiers have much disagreement on the SL2 data. The amount of disagreement can be used to measure the decision uncertainties, and is therefore useful for indicating incipient anomalies. However, for SL1 data that are close to the normal cluster, the above approach will become less effective. We find this is a common phenomenon in our empirical studies. A remedy to this problem is to increase the statistical power of the base learners by moving the decision boundaries towards the normal cluster. Another question is how to properly combine the anomaly scores from ensemble members into an uncertainty metric to inform decision making. We will analyze two widely used uncertainty metrics and compare their performance on detecting incipient anomalies.

We believe our proposed methods are a useful complement to the literature on multi-grade anomaly detection [19, 23, 24], specifically under cases where the available anomaly data for training are insufficient to cover the entire severity spectrum. In this paper, we give some caveats and an easy-to-use recipe for Machine Learning (ML) practitioners to develop ensemble anomaly detection models that can more effectively recognize incipient anomaly examples, and will provide recommendations to address the aforementioned issues that can help produce more effective ensemble models for anomaly detection applications. We summarize our contributions in this paper as follows:

- We show by experiments that incipient anomaly examples, when missing or underrepresented in the training distribution, can pose risks to popular supervised ML-based anomaly detection models such as Decision Trees (DTs) and Neural Networks (NNs). Ensemble methods are in general helpful in improving the detection performance of both supervised and unsupervised models on such incipient anomalies.
- We compare and analyze two commonly used uncertainty metrics for ensemble learning, one based on ensemble mean (mean) and the other based on ensemble variance (var). Our theoretical analysis shows that mean is more preferable to var.

The rest of this paper is organized as follows. We formulate the anomaly detection problem and provide necessary background knowledge in Sec. 2. Next in Sec. 3, we describe our methodology in details. The two datasets used in our empirical study are briefly described in Sec. 4, and in Sec. 5 we present our experimental results. In Sec. 6, we review related research topics found in the literature. We summarize the findings in this paper and discuss future work in Sec. 7.

2 BACKGROUND AND PROBLEM FORMULATION

Anomaly detection problem. We formulate the anomaly detection problem in a binary classification setting. An anomaly detection system aims at differentiating the fault conditions from the normal condition by monitoring the system state. Let \( z \in \{0, 1\} \) be the ground-truth label of system state \( x \in \mathbb{R}^d \), where \( z = 0 \) stands for the normal condition and \( z = 1 \) the anomaly condition. An anomaly detector is some rule, or function, that assigns (predicts) a class label \( \hat{z} \in \{0, 1\} \) to input \( x \).

Let \( X \) be the set of data points, and \( M \) be a model class of classification models. Suppose a classification model \( M \in M \) defines an anomaly score function \( s^M : X \to \mathbb{R} \) that characterizes how likely a data point corresponds to an anomaly state; a larger \( s^M(x) \) implies a higher chance of a data point \( x \) being an anomaly. The classifier’s decision on whether or not \( x \) corresponds to an
anomaly can be made by introducing a decision threshold $\tau^M$ to dichotomize the anomaly score $s^M(x)$. We can define the classifier’s predicted label $\hat{z} = 1 \{s^M(x) > \tau^M\}$, i.e. $M$ predicts $x$ to be an anomaly if and only if the anomaly score $s^M(x)$ is above the threshold $\tau^M$. For evaluating the accuracy of anomaly detection, we can define the False Negative Rate (FNR) and False Positive Rate (FPR) of the model $M$ on the test data distribution as follows:

$$\text{FNR}(s^M, \tau^M) = \mathbb{P} [\hat{z} = 0 \mid z = 1],$$

(1)

$$\text{FPR}(s^M, \tau^M) = \mathbb{P} [\hat{z} = 1 \mid z = 0].$$

(2)

Let $X^{\text{train}}$ be a subset of labeled data points for training. Ideally, our goal is to learn an anomaly score function $s^*$ by minimizing its classification error on $X^{\text{train}}$, and then decide a corresponding threshold $\tau$, such that $(s^*, \tau)$ can optimally trade off the FNR and the FPR on unseen test data.

Setting the detection threshold $\tau$. We leverage the prediction uncertainties given by ensemble learners to make uncertainty-informed decisions. Consider an ensemble $E$ comprising a diverse set of $K$ binary classifiers, $M^{(1)}, M^{(2)}, \ldots, M^{(K)}$, that have been trained for the same detection task. Let $z_i \in \{0, 1\}$ represent the ground-truth label of input $x_i$, and $\hat{y}_i^{(k)}$ denote the output of the $k$th classifier where $k \in \{1, \ldots, K\}$ and $\hat{y}_i^{(k)} \in [0, 1]$. By using a threshold $\tau$ to dichotomize the continuous output $\hat{y}_i^{(k)}$, each classifier $M^{(k)}$ produces a predicted class label $\hat{z}_i^{(k)}$ for input $x_i$.

As mentioned above, one always has to make a trade-off between FNR and FPR by setting an appropriate decision threshold $\tau$ (a.k.a. operating point). A simple approach is to directly set the decision threshold $\tau$ to a predefined value (e.g., 0.5); this is often not a bad approach if most data points are well separated and receive an anomaly score close to 0 or 1. However, such approach usually does not returns us a high-sensitivity classifier that satisfies a given FPR requirement. In real-practice, one often needs to decide a proper operating point on the Receiver Operating Characteristic (ROC) curve by taking FPR and FNR requirements into account. One way to do that is to set $\tau$ such that the FPR on the development set reaches a predefined level $q$. The rationale behind such scheme is to fix the FPR (type-1 errors) to a constant value on the development set while minimizing the number of false negatives (type-2 errors). Similar approaches are seen in other application domains. For example, in radar applications, this scheme is also known as Constant False Alarm Rate (CFAR) detection [31].

The decision scheme described above is illustrated in Fig. 2 as the baseline scheme. The goal is to come up with a proper $\tau$ that is used to identify positive examples. Under most cases, there will be false positives among the examples predicted as positive; however, these false positives are not the utmost concern if the FPR can be controlled to a low level. On the other hand, false negatives are anomalous instances mistaken as normal, which represents a more severe problem in anomaly detection. We propose utilizing decision uncertainty information from ensemble classifiers to identify potential false negatives in an uncertainty-informed decision scheme.

Uncertainty-informed anomaly detection. We consider an uncertainty-informed diagnostic scheme as an application of prediction uncertainties that fosters the collaboration between human and AI systems. In this scheme, an ML model is first used to screen the cases (operational data for industrial machines, medical images for humans, etc.). Cases diagnosed as positive will be referred to a human reviewer for further inspection, who will confirm the case as positive if she agrees with the ML model’s decision. The baseline scheme suffers from the problem that false negatives from the ML model’s diagnoses would never be reviewed by human diagnosticians. In an uncertainty-informed scheme, high-uncertainty negative examples will be identified and sent to human reviewers as well. The criterion used for picking out high-uncertainty examples does not have to be based on the
classifier confidence $\hat{y}$; in fact, we can use a variety of uncertainty metrics to be described below for ranking data examples by their associated uncertainties.

To identify false negatives in classification, we use an uncertainty metric $U$ to rank the negative examples\(^2\). The uncertainty metric $U: \mathbb{R}^K \rightarrow \mathbb{R}$ takes as input the ensemble predictions $\{\hat{y}_i^{(k)}\}$ on $x_i$, and outputs an uncertainty score $u(x_i) \doteq U(y_1^{(1)}, y_2^{(2)}, \ldots, y_K^{(K)})$. The interpretation of the uncertainty score $u(x_i)$ depends on the task. In our application, we seek to utilize prediction uncertainties to identify false negative decisions: a higher $u(x_i)$ indicates higher prediction uncertainty associated with $x_i$. In such situations, we may need human experts to join the decision process.

The uncertainty score $u(x_i)$ is a real-valued number, and to resolve a dichotomy between “uncertain” and “certain” we will need another threshold $\tilde{u}$. If $u(x_i) > \tilde{u}$ then $x_i$ is deemed an uncertain input example and otherwise a certain one. Once uncertain examples are identified, we will need external resources (e.g., human experts) to inspect them and determine their true labels; however, such external resources are often limited (e.g., due to budget constraints) so we need to determine a proper threshold $\tilde{u}$ so that the number of uncertain negatives is controlled. We determine $\tilde{u}$ by setting the uncertain negative ratio on the development set to a fraction $\theta$, where the uncertain negative ratio is the fraction of uncertain examples among those examples predicted as negative. Only the predicted negative examples that receive the highest uncertainty scores are deemed uncertain negatives. To evaluate how uncertain negatives overlap with the actual false negatives, we define the following performance measure.

**Definition 2.1 (False Negative Precision).** We define the false negative precision to be the fraction of false negative decisions among uncertain negative inputs, under given uncertainty metric $U$ and uncertain negative ratio $\theta$. Written in mathematical form,

$$FN\text{-}precision(U, q) = \frac{\left\{ x_i \mid i \in I_q^-, z_i = 1 \right\}}{|I^-_q|} \in [0, 1],$$

where $I^-_q$ is the index set of uncertain negative examples.

\(^2\)Examples that are classified as negative by a classification model, i.e. $\{x_i \mid \hat{z}_i = 0\}$. 

---

Fig. 2. Illustration showing the concepts in an uncertainty-informed decision framework.
The FN-precision metric can be interpreted as the ratio of identified uncertain examples being actual false negatives. The higher the FN-precision value, the fewer false alarms are likely to be raised by an algorithm that detects uncertain negatives. We can similarly define a “false negative recall” metric that measures the fraction of false negatives identified by the algorithm. However, in this paper we choose to directly report the total number of false negatives and the number of false negatives that are deemed “certain” by the evaluated uncertainty estimation algorithms, as we think it a more straightforward way to make the comparison. One of our goals in this paper is to rigorously analyze and compare two commonly used uncertainty metrics \( \text{mean} \) and \( \text{var} \), which will be detailed in the upcoming section. Formally, we seek the uncertainty metric \( U \) that maximizes FN-precision\((U, q)\).

3 METHODOLOGY

In the sequel, we describe our methodology of using ensemble learning for improving incipient anomaly detection. We first discuss several design considerations when constructing ensemble classifiers.

3.1 Creating Diversity among Base Learners in an Ensemble

Diversity is recognized as one of the key factors that contribute to the success of ensemble approaches [3]. As illustrated in Fig. 1b, the diversity among ensemble members is crucial for improved detection performance on out-of-distribution (o.o.d.) data instances. In our empirical study to be described later, we will employ bagging [2] to induce diversity among ensemble members. Bagging [2] (or bootstrap aggregation) is a classical approach for creating diversity among ensemble members. The core idea is to construct models from different training datasets using randomization. In the original bagging approach [2], a random subset of the training samples is selected for training each member classifier. A later variant, the so-called “feature bagging” (a.k.a. random subspace method [14]) selects a random subset of the features for training. One famous application of bagging in ML is the Random Forest (RF) model. In this study, we will use sample bagging in our experiments to induce diversity among ensemble classifiers.

3.2 Combining Base Models into an Ensemble

In ensemble analysis, one challenge is that the anomaly scores given by different models may not be directly comparable. This is known as the normalization issue [1]. This issue is more common with unsupervised or semi-supervised models, because the outputs from these models (e.g., the reconstruction errors from autoencoders) are often naturally unbounded. If the scores from different models are directly combined without normalization (e.g., by calculating the average or the maximum), models that give higher anomaly scores may be inadvertently favored [1]. The normalization issue is less concerning for supervised classification models that use a softmax layer to produce probability vectors whose values are bounded within the \([0, 1]\) interval; still, there are still concerns about whether or not these probability estimates are well calibrated (known as the calibration issue). For the ensemble supervised classifiers in this study, we assume minimal impacts from the calibration issues.

On top of the normalization issue, how to properly aggregate the (normalized) anomaly scores from models in an ensemble, known as the combination issue [1], constitutes another major challenge in ensemble analysis. Depending on how the base learners are combined into an ensemble detector, we can classify the combination scheme into a 1) hard voting or a 2) soft voting scheme. In hard voting schemes, each base learner predicts a binary label \( \hat{y} \in \{0, 1\} \) indicating whether an input example \( x \) is normal or not, while in soft voting schemes base learners outputs real-valued anomaly scores. In this work, we will mainly consider ensemble models made up of supervised classifiers,
and focus on how to properly obtain uncertainty estimates from the score vectors in order to better detect incipient anomalies.

### 3.3 Uncertainty Metrics for Ensemble Learners

Suppose we have \( m \) data points for testing, and they are organized into a design matrix \( \mathbf{X} \in \mathbb{R}^{m \times d} \). The outputs from the ensemble of detection models can be accordingly written as an \( m \times K \) matrix \( \hat{\mathbf{Y}} \), where \( K \) is the ensemble size. Note that entries in matrix \( \hat{\mathbf{Y}} \) can either take discrete values from \( \{0, 1\} \) (in a hard voting scheme) or take continuous values from \([0, 1]\) (in a soft voting scheme), depending on the nature of the underlying base learners. Using superscripts to differentiate ensemble members and subscripts to differentiate data points, we can denote the rows and columns of matrix \( \hat{\mathbf{Y}} \) as follows

\[
\hat{\mathbf{Y}} \equiv \begin{bmatrix}
\hat{y}_1^{(1)} & \hat{y}_2^{(1)} & \cdots & \hat{y}_m^{(1)} \\
\hat{y}_1^{(2)} & \hat{y}_2^{(2)} & \cdots & \hat{y}_m^{(2)} \\
\vdots & \vdots & \ddots & \vdots \\
\hat{y}_1^{(K)} & \hat{y}_2^{(K)} & \cdots & \hat{y}_m^{(K)}
\end{bmatrix}
\]

(4)

where each \( \hat{y}_i^{(k)} \) represents the predictions from the \( k \)th single learner \( (k = 1, 2, \ldots, K) \) on the \( m \) data points, and each \( \hat{y}_i = \begin{bmatrix} \hat{y}_i^{(1)} \ 
\hat{y}_i^{(2)} \cdots \hat{y}_i^{(K)} \end{bmatrix} \) represents the \( K \) predictions from the ensemble learner on \( x_i \).

To come up with an uncertainty estimate for \( x_i \), we calculate \( U(\hat{y}_i^{(1)}, \hat{y}_i^{(2)}, \cdots, \hat{y}_i^{(K)}) \) using \( U \) as the uncertainty metric. A number of metrics have been proposed in literature for estimating the prediction uncertainties of ensemble learners. In Lakshminarayanan et al. [20], the metrics are broadly classified into two categories: confidence-based and disagreement-based metrics. The former category is designed to capture the consensus of the individual learners in an ensemble, while the latter aims to measure the degree of disagreement among their predictions; however, the two seemingly unrelated goals can have a significant overlap. In this paper, we propose a rigorous categorization for these uncertainty metrics depending on their mathematical forms to unveil their differences and to enable further analyses. Some metrics (hereinafter referred to as type-1 metrics) rely only on the ensemble output \( \hat{y}_i^e \), while others (referred to as type-2 metrics) take all single learner’s outputs into account. Type-1 metrics use the ensemble output \( \hat{y}_i^e \) to compute the confidence level, without the need to know what the individual predictions are. A negative aspect of these metrics is that the disagreement among individual learners can be hidden beneath the ensemble output \( \hat{y}_i^e \).

#### Confidence Gap Metric (\textit{MEAN})

An intuitive metric that measures the confidence of a classifier on input \( x \) is to see how close the prediction \( \hat{y}_i^e \) is to the decision threshold \( \tau^e \). Here the superscripts in \( \hat{y}_i^e \) and \( \tau^e \) signify values associated with an ensemble classifier; in the special case where \( K = 1 \), the ensemble classifier degenerates to a single learner classifier. The smaller the gap \( |\hat{y}_i^e - \tau^e| \) is, the higher the uncertainty with \( x_i \). Since we prefer the convention that larger function values of \( u^\text{MEAN}(x_i) \) corresponds to larger uncertainties, we define the uncertainty score under the margin metric can be formulated as

\[
u^\text{MEAN}(x_i) = 1 - |\hat{y}_i^e - \tau^e|,
\]

(5)

where a constant 1 is added to the definition so that the uncertainty value \( u^\text{MEAN}(x) \) is always positive. Since the ensemble prediction \( \hat{y}_i^e \) is obtained by taking the average of the individual outputs of classifiers in the ensemble, we will hereinafter refer to this metric as \textit{MEAN}.
**Binary Cross-Entropy Metric (Entropy).** The binary cross-entropy \( u_{\text{ENTROPY}} \) as a function of \( x_i \) takes the form

\[
u_{\text{ENTROPY}}(x_i) \doteq - \left[ \hat{y}_i^e \log \hat{y}_i^e + (1 - \hat{y}_i^e) \log (1 - \hat{y}_i^e) \right]
\]  

Entropy is equivalent to mean when the decision threshold \( \tau^e = 0.5 \). It can be easily proved that when \( \tau^e = 0.5 \),

\[
u_{\text{MEAN}}(x_i) > \nu_{\text{MEAN}}(x_j) \iff u_{\text{ENTROPY}}(x_i) > u_{\text{ENTROPY}}(x_j).
\]  

In other words, when \( \tau = 0.5 \) the rankings assigned by \( u_{\text{ENTROPY}} \) and \( u_{\text{MEAN}}(x) \) to the data points are the same. Since we identify uncertain examples by finding the top-ranked data points, \( u_{\text{ENTROPY}} \) and \( u_{\text{MEAN}}(x) \) are equivalent. The \( u_{\text{ENTROPY}} \) metric can be useful for evaluating decision uncertainties when no decision threshold is a priori assigned.

Comparing to the type-1 metrics described above, type-2 metrics have the potential to give a more comprehensive characterization of the individual predictions (e.g., the disagreement among \( \hat{y}_i^{(1)}, \hat{y}_i^{(2)}, \ldots, \hat{y}_i^{(K)} \)). The following two existing type-2 metrics that are often used in literature, focus on quantifying the disagreement among individual learners in an ensemble and for this reason, may be able to address the shortcomings of type-1 metrics.

**Variance Metric (var).** The variance (or standard deviation) metric \([17, 21]\) measures how spread out the individual learners’ predictions from the ensemble prediction \( \hat{y}_i^e \). The uncertainty score of input \( x_i \) based on sample variance can be written as

\[
u_{\text{VAR}}(x_i) \doteq \frac{1}{K-1} \sum_{k=1}^{K} \left[ \hat{y}_i^{(k)} - \hat{y}_i^e \right]^2
\]  

**Kullback–Leibler (KL) Divergence Metric (KL).** Similar to the variance metric, the KL divergence metric \([10]\) measures the deviation of individual learner’s predictions from the ensemble output \( \hat{y}_i^e \). The uncertainty score \( s_{\text{KL}}(x_i) \) of input \( x_i \) under the KL divergence metric can be written as

\[
u_{\text{KL}}(x_i) \doteq \frac{1}{K} \sum_{k=1}^{K} D_{\text{KL}} \left( \hat{y}_i^{(k)} \parallel \hat{y}_i^e \right) = \sum_{k=1}^{K} \hat{y}_i^{(k)} \log \frac{\hat{y}_i^{(k)}}{\hat{y}_i^e}.
\]  

A problem with var and kl is that they focus mainly on the disagreement among ensemble predictions but do not take in consideration the value of \( \hat{y}_i^e \). Consider a scenario where the all ensemble members predict a probability of 0.5. Both var and kl will produce an uncertainty score of 0 and thus will not be able to capture any decision uncertainties; in fact, this case where all learners give an output of 0.5 is highly uncertain. Next, we will compare two representative uncertainty metrics, mean and var, from a theoretical perspective.

### 3.4 Theoretical Analysis on Uncertainty Metrics mean and var

To model how different classifiers will respond to a given input \( x_i \), we assume that the prediction \( \hat{y}_i^{(k)} \) from classifier \( \mathcal{M}^{(k)} \) is sampled from a beta distribution \( \mathcal{B}(\alpha_i, \beta_i) \) that is characterized by two parameters by \( \alpha_i \) and \( \beta_i \). We further assume that \( \alpha_i + \beta_i \) is fixed to the same constant value for all \( i \)'s. Under this assumption, the SL of the case represented by \( x_i \) can be characterized by a single parameter \( \alpha_i \), easing further analysis. The larger the value of \( \alpha_i \), the more severe the case of \( x_i \) is. When \( \alpha_i \) and \( \beta_i \) are close, the case is ambiguous as the distribution shifts towards being symmetric (i.e. signifying much disagreement) rather than being one-sided.

The main theoretical contribution of this paper is presented in the following theorem, which implies that if \( x_i \) is more likely to be positive than \( x_j \), then for ensemble learners of fixed size, the
upper bound on the probability of \( s_{\text{mean}} \) making a wrong decision is lower. In other words, \( s_{\text{mean}} \) is likely to be a more robust measure than \( s_{\text{var}} \).

**Theorem 3.1.** Consider inputs \( x_i, x_j \), with \( y_i \sim \mathcal{B}(\alpha_i, \beta_i) \), \( y_j \sim \mathcal{B}(\alpha_j, \beta_j) \), and \( \alpha_i + \beta_i = \alpha_j + \beta_j \). Let \( \Delta_{ij}(s) := \mathbb{E}[s(x_j) - s(x_i)] \) where \( s() \) denotes an uncertainty score estimated from \( K \) i.i.d. ensemble learners. If \( \alpha_i < \alpha_j \leq \beta_j \), then \( \Delta_{ij}(s_{\text{mean}}) > \Delta_{ij}(s_{\text{var}}) > 0 \). Furthermore, it holds that

\[
\Pr\left(s_{\text{mean}}(x_i) > s_{\text{mean}}(x_j)\right) = O\left(\frac{1}{K\Delta_{ij}(s_{\text{mean}})}\right) \quad \text{and} \quad \Pr\left(s_{\text{var}}(x_i) > s_{\text{var}}(x_j)\right) = O\left(\frac{1}{K\Delta_{ij}(s_{\text{var}})}\right).
\]

The choice of uncertainty metric \( U \) determines how examples are ranked and therefore affects the detection performance of false negatives. We expect the final ranking negative examples due to the uncertainty metric \( U \) matches the true severity ranking given by \( \alpha_i \). Taking a microscopic view into the ranking process, we consider two negative examples \( x_i \) and \( x_j \), and assume \( x_i \) represents a less severe case than \( x_j \). Under the above beta distribution assumption, we will have \( \alpha_i < \alpha_j \leq \beta_j \).

Our theoretical analysis will focus on the chance that \( x_i \) (the less ambiguous or more normal case) is considered more uncertain than \( x_j \) (the more ambiguous case). If the following theorem holds, then those correctly ranked by \( \text{var} \) are also likely to be correctly ranked by \( \text{mean} \), indicating that \( \text{mean} \) is a preferable uncertainty metric to \( \text{var} \).

**Lemma 3.2.** Consider two inputs \( x_i, x_j \) with uncertainty score \( s(x_i) \) and \( s(x_j) \) estimated from \( K \) i.i.d. ensemble learners, and denote by \( \Delta_{ij}(s) := \mathbb{E}[s(x_j) - s(x_i)] \) the difference of expected uncertainty score. If \( \Delta_{ij}(s) > 0 \), then \( \Pr\left(s(x_i) > s(x_j)\right) = O\left(\frac{\Var(s(x_j)) + \Var(s(x_i))}{\Delta_{ij}(s)}\right) \).

The proof is provided below. Intuitively, Lemma 3.2 states that if input \( x_j \) is more uncertain than \( x_i \) w.r.t. the expected uncertainty \( \mathbb{E}[s()] \), then the probability of the sample uncertain measure \( s \) making a wrong decision is bounded. Based on such result, we establish the following error bounds for uncertainty metrics \( \text{mean} \) and \( \text{var} \).

**Proof.** Let \( Z = s(x_j) - s(x_i) \) be a random variable, where \( s(x_i) \) and \( s(x_j) \) denotes the uncertainty score of \( x_i \) and \( x_j \) estimated from \( K \) i.i.d. ensemble learners. Therefore \( \Delta_{ij}(s) = \mathbb{E}[Z] > 0 \). By Chebyshev’s Inequality, we obtain

\[
\Pr\left(\left|Z - \mathbb{E}[Z]\right| \geq \Delta_{ij}(s)\right) \leq \frac{\Var(Z)}{\Delta_{ij}^2(s)},
\]

which implies that

\[
\Pr\left(Z - \mathbb{E}[Z] \leq -\Delta_{ij}(s)\right) = \Pr\left(Z - \Delta_{ij}(s) \leq -\Delta_{ij}(s)\right) = \Pr\left(s(x_j) - s(x_i) \leq 0\right) \leq \frac{\Var(Z)}{\Delta_{ij}^2(s)}.
\]

Further noticing that \( \Var(Z) = \Var(s(x_j) - s(x_i)) = \Var(s(x_j)) + \Var(s(x_i)) \), we conclude that

\[
\Pr\left(s(x_i) > s(x_j)\right) = O\left(\frac{\Var(s(x_j)) + \Var(s(x_i))}{\Delta_{ij}^2(s)}\right),
\]

which completes the proof. \( \square \)

Based on Lemma 3.2, below we provide the proof of Theorem 3.1.

**Proof.** To prove the first statement, i.e. \( \Delta_{ij}(s_{\text{mean}}) > \Delta_{ij}(s_{\text{var}}) > 0 \), we consider the following properties of a beta distribution \( \mathcal{B}(\alpha, \beta) \).

\[
\mu = \frac{\alpha}{\alpha + \beta}, \quad \sigma = \frac{\alpha \beta}{(\alpha + \beta)^2(1 + \alpha + \beta)} = \frac{\mu(1 - \mu)}{1 + \alpha + \beta}.
\]
where $\mu_i$ and $\sigma_i$ respectively represent the mean and variance of the beta distribution $\mathcal{B}(\alpha_i, \beta_i)$.

Let $\alpha_i + \beta_i = \alpha_j + \beta_j = c$. Since $\alpha_i < \alpha_j \leq \beta_j$, we know

$$\mu_i = \frac{\alpha_i}{\alpha_i + \beta_i} < \frac{\alpha_j}{\alpha_j + \beta_j} = \mu_j \leq \frac{1}{2}, \quad \sigma_i = \frac{\mu_i(1 - \mu_i)}{1 + \alpha_i + \beta_i} < \frac{\mu_j(1 - \mu_j)}{1 + \alpha_j + \beta_j} = \sigma_j$$

Therefore, we have

$$\Delta_{ij}(s_{\text{MEAN}}) = \mathbb{E}\left[ s_{\text{MEAN}}(x_j) - s_{\text{MEAN}}(x_i) \right] = \mu_j - \mu_i > 0,$$
$$\Delta_{ij}(s_{\text{VAR}}) = \mathbb{E}\left[ s_{\text{VAR}}(x_j) - s_{\text{VAR}}(x_i) \right] = \sigma_j - \sigma_i > 0.$$ 

Furthermore, notice that

$$\Delta_{ij}(s_{\text{VAR}}) = \frac{\mu_j(1 - \mu_j) - \mu_i(1 - \mu_i)}{1 + c} < \mu_j(1 - \mu_j) - \mu_i(1 - \mu_i) = \mu_j - \mu_i - (\mu_j^2 - \mu_i^2) < \Delta_{ij}(s_{\text{MEAN}}),$$

which proves the first statement of Theorem 3.1.

To prove the second statement, i.e., to provide an upper bound on the errors of $s_{\text{MEAN}}$ and $s_{\text{VAR}}$, we plug in the definition of $s_{\text{MEAN}}$ and $s_{\text{VAR}}$ to Lemma 3.1:

$$\Pr\left( s_{\text{MEAN}}(x_i) > s_{\text{MEAN}}(x_j) \right) = O\left( \frac{\text{Var}(s_{\text{MEAN}}(x_i)) + \text{Var}(s_{\text{MEAN}}(x_j))}{\Delta_{ij}(s_{\text{MEAN}})} \right) \overset{(a)}{=} O\left( \frac{\sigma_j + \sigma_i}{K\Delta_{ij}(s_{\text{MEAN}})} \right) = O\left( \frac{1}{K\Delta_{ij}(s_{\text{MEAN}})} \right)$$

where step (a) is due to $\text{Var}(s_{\text{MEAN}}(x_i)) = \sigma_i/n$. Similarly,

$$\Pr\left( s_{\text{VAR}}(x_i) > s_{\text{VAR}}(x_j) \right) = O\left( \frac{\text{Var}(s_{\text{VAR}}(x_i)) + \text{Var}(s_{\text{VAR}}(x_j))}{\Delta_{ij}(s_{\text{VAR}})} \right) \overset{(b)}{=} O\left( \frac{1}{K\Delta_{ij}(s_{\text{VAR}})} \right)$$

Here, step (b) is due to the variance of sample variance $\text{Var}(s_{\text{VAR}}(x_i)) = \frac{1}{K}(\mu_4 - \sigma^2(x_i)) + O(n^{-2}) = O\left( \frac{1}{K} \right)$ [4] where $\mu_4$ is the Kurtosis of the Beta distribution $\mathcal{B}(\alpha_i, \beta_i)$.

A direct corollary of the above theorem states that under infinite ensemble size, using either $\text{MEAN}$ or $\text{VAR}$ as the uncertainty metric does not make a difference.

**Corollary 3.3.** If the sample size is infinite, then under the conditions of Theorem 3.1, we have $s_{\text{MEAN}}(x_i) < s_{\text{MEAN}}(x_j) \Leftrightarrow s_{\text{VAR}}(x_i) < s_{\text{VAR}}(x_j)$.

### 4 DATASET DESCRIPTIONS

#### 4.1 ASHRAE RP-1043 Chiller Dataset

We used the ASHRAE RP-1043 Dataset [5] (hereinafter referred to as the “chiller dataset”) to examine the proposed ensemble approach for detecting incipient anomalies. In the chiller dataset, sensor measurements of a typical cooling system—a 90-ton centrifugal water-cooled chiller—were recorded under both anomaly-free and various fault conditions. In this study, we included the six faults (FWE, FWC, RO, RL, CF, NC) used in our previous study [17] as the anomaly (positive) class; each fault was introduced at four levels of severity (SL1–SL4, from the slightest to the severest). We consider SL3 and SL4 cases as severe faults, and SL1 and SL2 cases as incipient faults. For feature selection, we also followed our previous study [17] and used the sixteen key features therein for training our models. To give the readers an intuitive view about the distribution of the chiller
data, we employed the Linear Discriminant Analysis (LDA) algorithm to reduce the data into two dimensions, and visualized part of the reduced-dimension data in Fig. 1a described earlier. We can observe a general trend in the visualization: data points will deviate further away from the normal cluster when the corresponding fault develops into a higher SL.

4.2 Kaggle Diabetic Retinopathy Dataset

As another case study, we examined the efficacy of our proposed approach on a medical image dataset of Diabetic Retinopathys (DRs) diseases. DR is a common complication of the diabetic disease and the leading cause of blindness in the working-age population of the developed world [11]. The Kaggle-DR dataset [6] (hereinafter referred to as the “DR dataset”) comprises 88,702 high resolution images. Similar to the above-mentioned chiller faults, the presence of DR is also rated into five different SLs: no-DR (SL0), mild (SL1), moderate (SL2), severe (SL3), and proliferate (SL4), as illustrated in Fig. 4. Again, SL3 and SL4 are considered severe anomalies. It is worthy to note one key difference between the two datasets: the SL1 cases in the DR dataset are considered non-referable disease type and thus belong to the negative class.

4.3 Partitioning the Datasets

We divided each dataset into a development set and a test set. The test set can be further divided into two parts; one contains only the normal data (SL0) and the non-incipient anomalies (SL3 & 4), the other containing only the incipient anomalies; see Fig. 3 for an illustration. All five SLs were present in the development set data. To model how the availability of incipient anomaly data affected the detection performance, we introduce a parameter, the incipient anomaly ratio $\rho$, to control the proportion of incipient anomaly data that enters the development set. In our experiment, we tested $\rho = 0, 0.2, 0.4, 0.6, 0.8, 1.0$. It is worthy to note that when $\rho = 0$, no incipient anomaly data appeared in the development set; in other words, the incipient anomaly data became o.o.d. because they were not present at training time. We specifically included this scenario to see if the models can learn useful knowledge from only non-incipient anomalies that is useful for identifying incipient anomalies. Further details about the two datasets as well as the preprocessing steps will be given in the appendix at the end of this paper.
5 EXPERIMENTAL STUDY

5.1 Experimental Setup

Real-world ML practitioners perform extensive model selection to search for models on the development set data, and pick those that are more likely to perform well on test sets. We employed a similar workflow in our empirical study. For each model class under study, we swept over a wide range of hyperparameter settings, picked out a set of best-performing hyperparameters, and assess whether or not our proposed ensemble method could deliver consistent performance improvement compared to the baseline scenarios.

We experimented using DT and NN base learners to construct ensembles for the chiller dataset where sensor data assume a tabular form. For the DR dataset, we trained multiple Convolutional Neural Network (CNN) models of different architectures and data augmentation settings, and combined them into ensembles. Each ensemble model only consisted of base learners of the same type. In our empirical study, we evaluated ensembles of four different sizes: 5, 10, 15 and 25, and compared their performance to the single learner case. Further implementation details will be deferred to the appendix of this paper. Next, we will report the experimental results.

5.2 Detection Performance of Ensemble Classifiers

5.2.1 False Negative Rate (FNR). We first report the detection performance of the trained ensembles on the two datasets in terms of FNR. We examine the FNR for both incipient and non-incipient anomalies under different settings of FPR percentile $q$ and the incipient anomaly ratio $\rho$, and show the results as box plots in Fig. 5. We only show the FNR results on incipient anomaly cases for single learners (left panel) and for ensemble learners of size 25 (right panel); the FNR for non-incipient anomalies are all close to zero, which indicates near-perfect classification performance between SL0 (normal conditions) and SL3 & SL4 (non-incipient anomalies). The results for non-incipient anomalies are not displayed here due to limited space. By comparing the two cases ($K = 1$ vs. $K = 25$), we can immediately see performance improvement for ensemble learners over single learners. In addition, we can observe a decreasing trend in FNR with increasing $q$, which indicates that more incipient anomalies can be detected when we lower the detection threshold $\tau$; in other words, more incipient anomalies can be detected when the classifiers are working at more sensitive operating points.
5.2.2 Remaining False Negatives. The next performance index we evaluate is the number of remaining false negatives after applying uncertainty estimation. The numbers of remaining false negatives are obtained by assuming that all identified uncertain false negatives will receive corrected labels. We are interested in knowing the number of remaining false negatives because these are mistakes that the uncertainty estimation techniques fail to identify. We visualize the performance variations of the trained models for the two datasets as box plots in Fig. 6 and in Fig. 7, respectively.

As displayed in the plots, besides **mean** and **var** we also included two other scenarios, **baseline** and **mean+var**, that respectively set the lower bound and the upper bound of performance of **mean** and **var**. Under **baseline**, no uncertainty information from output probabilities is utilized, i.e. $q = 0$. **mean+var** is a hypothetical uncertainty metric where the uncertain examples identified by **mean+var** are the union of the two sets of uncertain examples identified by **mean** and by **var**, not subject to the constraint imposed by $q$; see Fig. 2 for an illustration. Therefore, it is at least as good as **mean** or **var**. If **mean** and **var** do not have much overlapping, **mean+var** will identify
many more false negatives than either of them alone; however, we can see from Fig. 6 and Fig. 7 that this is not the case. The results given by mean+var do not have much improvement over those given by mean, indicating that many of the false negatives identified by var are also captured by mean, matching the expectation of Theorem 3.1.

An immediate observation from Figs. 6 & 7 is that ensemble learning can achieve substantial performance improvement even for small ensemble sizes \((K = 5)\). For \(K > 5\), we can still see significant improvement when \(K\) grows larger for tree ensembles; however, for NN ensembles the marginal improvement from increasing ensemble sizes is smaller, which is probably due to the fact that individual DT classifiers are relatively weak compared to individual NN classifiers. By comparing the performance of mean and that of var in the plots, we can see that mean leads to fewer remaining false negatives in general; in other words, the mean uncertainty metric can identify more false negatives than the var metric can.

5.2.3 False Negative Precision (FN-precision). Although the above analysis shows that mean compared to var can identify more false negatives among incipient anomalies, it is not sufficient to show that mean is more preferable to var because the increased number of corrected false negatives may simply be a consequence of more uncertain negatives being identified; in an extreme scenario, if all negative data points are marked as uncertain negatives, then all false negatives can be corrected. Therefore, we introduce the FN-precision metric to measure how precisely each model can identify the false negatives. As can be seen from Fig. 8, mean again outperforms var in terms of FN-precision.

5.3 Detection Performance of One-Class Classifiers
As a comparative study, we also experimented using One-Class Support Vector Machine (OC-SVM), a popular one-class model for semi-supervised and unsupervised learning, to learn a boundary of the normal data points (i.e., the inliers) that can be used to separate them from the outliers for the chiller dataset. We did not experimented applying OC-SVM on the DR dataset as it does apply to image data. Again, we conducted a grid search over various hyperparameter settings and picked out the best-performing models.

In Fig. 9, we visualize the performance of OC-SVMs ensembles of three different sizes \(K = 1, 5, 25\), and show how the detection performance in terms of FNR varies with the FPR percentile \(q\). As with other learners for the chiller dataset, we used sample bagging to induce diversity among ensemble OC-SVM learners. The experimental results for ensemble learners, however, did not demonstrate much improvement over the single learner cases. By comparing the results for OC-SVM to those for DT and NN ensembles, we can see that OC-SVM gives inferior detection performance for both incipient and non-incipient anomalies. A detailed discussion on OC-SVM and other one-class methods (e.g., autoencoders) is beyond the scope of this paper. We believe there are also challenges ahead in applying one-class methods to anomaly detection, which will become promising directions for future research.

5.4 Validation of the Beta Distribution Assumption
In our theoretical analysis in Sec. 3.4, we make an assumption that the individual predictions in an ensemble learner assume a Beta distribution \(B(\alpha, \beta)\) where \(\alpha + \beta\) is held constant. We performed some observational studies to validate this assumption, as displayed by the five examples in Fig. 4. Additional examples will be given in the appendix.
Out-of-distribution Input Detection and Uncertainty Estimation. In recent years, a number of research papers [9, 20] related to the detection of o.o.d. data appeared in literature, especially in the deep learning community that has shown a strong and growing interest in utilizing ensemble methods in supervised learning to estimate the uncertainty behind the decisions on data points. Lakshminarayanan et al. [20] proposed using random initialization and random shuffling of training examples to diversify base learners of the same network architecture. Gal and Ghahramani proposed using MC-dropout [9] to estimate a network’s prediction uncertainty by using dropout not only at

### 6 RELATED WORK

Using Ensemble Classifiers to Detect Incipient Anomalies

---

Fig. 6. Box plots showing the number of certain false negatives (incipient anomalies wrongly classified as negative) after the rest are identified by uncertainty estimation for the chiller dataset. Results for \( \rho = 0, 0.2, 1.0 \) are displayed respectively in the left, the middle and the right columns.

(a) DT + non-incipient \((\rho = 0)\)  (b) DT + non-incipient \((\rho = 0.2)\)  (c) DT + non-incipient \((\rho = 1.0)\)

(d) DT + incipient \((\rho = 0)\)  (e) DT + incipient \((\rho = 0.2)\)  (f) DT + incipient \((\rho = 1.0)\)

(g) NN + non-incipient \((\rho = 0)\)  (h) NN + non-incipient \((\rho = 0.2)\)  (i) NN + non-incipient \((\rho = 1.0)\)

(j) NN + incipient \((\rho = 0)\)  (k) NN + incipient \((\rho = 0.2)\)  (l) NN + incipient \((\rho = 1.0)\)
training time but also at test time. By sampling a dropout model $M$ using the same input for $T$ times, we can obtain an ensemble of prediction results with $T$ individual probability vectors. The dropout technique provides an inexpensive approximation to training and evaluating an ensemble of exponentially many similar yet different neural networks.

Although promising results from these ensemble approaches have been demonstrated on certain types of o.o.d. data such as dataset shift and unseen/unknown classes [20], it is difficult to evaluate their effectiveness in general, because the o.o.d. part of the world is obviously much “larger” than its in-distribution counterpart and is presumably much harder to analyze. In contrast, our work, although using similar algorithms to those on o.o.d. detection, still embraces a closed-world assumption and restricts the focus to incipient anomalies—a special type of data distribution that has a close connection to the training distribution. We speculate that some knowledge necessary for detecting incipient anomalies is already entailed in the training data, thus making the detection of incipient anomalies possible with supervised methods.

Model calibration. Another relevant line of work aims to produce good probability outputs using model calibration techniques [12, 27], e.g. temperature scaling [12]. The calibration techniques are typically applied in a post-processing manner; in other words, a calibration method learns a transformation that is applied to a model’s uncalibrated output probabilities, without affecting the parameters (weights) of the original model itself. Although good confidence measures are important in many other fields, we are skeptical about the role of model calibration in the context of anomaly detection. By design, calibration methods should only adjust the probability values without affecting the ranking among data points. Therefore, if a calibration method results in an isotonic transformation (as are popular methods), the rankings will not change. Nor will the detection decisions.
We show in this paper that, incipient anomalies can pose critical challenges to supervised anomaly detection systems built upon ML techniques, especially under situations where incipient anomalies are absent from the training data. The resulting ML models can easily mistake incipient anomalies for normal ones, which can lead to costly consequences if the ideal time for intervention or treatment is missed. To address this challenge, we study how to exploit the uncertainty information from ensemble learners to identify incipient anomalies that are potentially wrongly classified. The three main takeaways from this study are summarized as follows:

- Without sacrificing the detection performance on non-incipient anomalies, we can improve the classifier’s performance on incipient anomalies by using models of higher sensitivity; this can be done by tuning down the a classifier’s detection threshold $\tau$.
- The detection performance on incipient anomalies can be greatly improved by incorporating some incipient anomaly data, even in small amount, into the training distribution (i.e. the development set).
- $\text{MEAN}$ is a more preferable uncertainty metric to $\text{VAR}$, as proved by our theoretical analysis and shown by our empirical results.

7 CONCLUSIONS AND FUTURE WORK

We show in this paper that, incipient anomalies can pose critical challenges to supervised anomaly detection systems built upon ML techniques, especially under situations where incipient anomalies are absent from the training data. The resulting ML models can easily mistake incipient anomalies for normal ones, which can lead to costly consequences if the ideal time for intervention or treatment is missed. To address this challenge, we study how to exploit the uncertainty information from ensemble learners to identify incipient anomalies that are potentially wrongly classified. The three main takeaways from this study are summarized as follows:

- Without sacrificing the detection performance on non-incipient anomalies, we can improve the classifier’s performance on incipient anomalies by using models of higher sensitivity; this can be done by tuning down the a classifier’s detection threshold $\tau$.
- The detection performance on incipient anomalies can be greatly improved by incorporating some incipient anomaly data, even in small amount, into the training distribution (i.e. the development set).
- $\text{MEAN}$ is a more preferable uncertainty metric to $\text{VAR}$, as proved by our theoretical analysis and shown by our empirical results.

Fig. 8. Box plots showing the false negative precision metric for ensemble classifiers ($K = 5, 10, 15, 25$) under different settings of the FPR percentile $q$ for the two datasets. Different colors indicate performance indices given by $\text{MEAN}$ and $\text{VAR}$ for the incipient and the non-incipient data.
(a) Non-incipient anomalies: $K = 1$  
(b) Non-incipient anomalies: $K = 5$  
(c) Non-incipient anomalies: $K = 25$  

(d) Incipient anomalies: $K = 1$  
(e) Incipient anomalies: $K = 5$  
(f) Incipient anomalies: $K = 25$

Fig. 9. The performance of OC-SVMs classifiers on non-incipient anomalies (top panel) and incipient anomalies (bottom panel). Box plots for ensembles of three different sizes $K = 1, 5, 25$ are displayed.

The three recommendations above are complementary and can lead to better results when applied together. It is worthy to note that in this paper we mainly focus on supervised ML models and their ensembles. One-class methods such as OC-SVMs and autoencoders are also promising and interesting directions to explore, which will constitute our future work.

**REFERENCES**

[1] Charu C Aggarwal. 2013. Outlier ensembles: position paper. *ACM SIGKDD Explorations Newsletter* 14, 2 (2013), 49–58.

[2] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.

[3] Gavin Brown, Jeremy Wyatt, Rachel Harris, and Xin Yao. 2005. Diversity creation methods: a survey and categorisation. *Information Fusion* 6, 1 (2005), 5–20.

[4] Eunghchum Cho and Moon Jung Cho. 2008. Variance of sample variance. *Section on Survey Research Methods—JSM* 2 (2008), 1291–1293.

[5] MC Comstock and JE Braun. 1999. Development of analysis tools for the evaluation of fault detection and diagnostics in chillers. ASHRAE Research Project RP-1043. *American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., Atlanta. Also, Report HL* (1999), 99–20.

[6] Jorge Cuadros and George Bresnick. 2009. EyePACS: an adaptable telemedicine system for diabetic retinopathy screening. *Journal of diabetes science and technology* 3, 3 (2009), 509–516.

[7] Min Du, Zhi Chen, Chang Liu, Rajvardhan Oak, and Dawn Song. 2019. Lifelong Anomaly Detection Through Unlearning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1283–1297.

[8] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1285–1298.

[9] Yarin Gal. 2016. Uncertainty in deep learning. *University of Cambridge* (2016).

[10] Jacob Goldberger, Shiri Gordon, and Hayit Greenspan. 2003. An efficient image similarity measure based on approximations of KL-divergence between two Gaussian mixtures. In *nulL. IEEE*, 487.

[11] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. 2016. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama* 316, 22 (2016), 2402–2410.

[12] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning—Volume 70*. JMLR. org, 1321–1330.
[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. 770–778.

[14] Tin Kam Ho. 1998. The random subspace method for constructing decision forests. IEEE transactions on pattern analysis and machine intelligence 20, 8 (1998), 832–844.

[15] Ling Huang, XuanLong Nguyen, Minos Garofalakis, Michael J Jordan, Anthony Joseph, and Nina Taft. 2007. In-network PCA and anomaly detection. In Advances in Neural Information Processing Systems. 617–624.

[16] Baihong Jin, Yufeng Chen, Dan Li, Kameshwar Poolla, and Alberto L. Sangiovanni-Vincentelli. 2019. A One-Class Support Vector Machine Calibration Method for Time Series Change Point Detection. 2019 IEEE International Conference on Prognostics and Health Management (ICPHM) (2019), 1–5.

[17] Baihong Jin, Dan Li, Seshadhri Srinivasan, See-Kiong Ng, Kameshwar Poolla, and Alberto Sangiovanni-Vincentelli. 2019. Detecting and diagnosing incipient building faults using uncertainty information from deep neural networks. In 2019 IEEE International Conference on Prognostics and Health Management (ICPHM). IEEE, 1–8.

[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. CoRR abs/1412.6980 (2015).

[19] Jonathan Krause, Varun Gulshan, Ehsan Rahimy, Peter Karth, Kasumi Widner, Greg S Corrado, Lily Peng, and Dale R Webster. 2018. Grader variability and the importance of reference standards for evaluating machine learning models for diabetic retinopathy. Ophthalmology 125, 8 (2018), 1264–1272.

[20] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In Advances in Neural Information Processing Systems. 6402–6413.

[21] Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. 2017. Leveraging uncertainty information from deep neural networks for disease detection. Scientific reports 7, 1 (2017), 17816.

[22] Dan Li. 2017. Fault detection and diagnosis for chillers and AHUs of building ACMV systems. Ph.D. Dissertation.

[23] Dan Li, Guoqiang Hu, and Costas Spanos. 2016. A data-driven strategy for detection and diagnosis of building chiller faults using linear discriminant analysis. Energy and Buildings 128 (2016), 519–529.

[24] Dan Li, Yuxun Zhou, Guoqiang Hu, and Costas J Spanos. 2016. Fault detection and diagnosis for building cooling system with a tree-structured learning method. Energy and Buildings 127 (2016), 540–551.

[25] Sébastien Marcel and Yann Rodriguez. 2010. Torchvision the Machine-Vision Package of Torch. In Proceedings of the 18th ACM International Conference on Multimedia (Firenze, Italy) (MM ’10). Association for Computing Machinery, New York, NY, USA, 1485–1488. https://doi.org/10.1145/1873951.1874254

[26] Comstock MC and Braun JE. 2002. Fault detection and diagnostic (FDD) requirements and evaluation tools for chillers. ASHRAE (2002).

[27] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In Proceedings of the 22nd international conference on Machine learning. 652–652.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011), 2825–2830.

[30] Luis Perez and Jason Wang. 2017. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. ArXiv abs/1712.04621 (2017).

[31] Mark A Richards. 2005. Fundamentals of radar signal processing. Tata McGraw-Hill Education.

[32] Mayu Sakurada and Takehisa Yairi. 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis. ACM, 4.

[33] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014).

[34] Yingshui Tan, Bailong Jin, Alexander Nettekoven, Yuxin Chen, Yisong Yue, Ufuk Topcu, and Alberto Sangiovanni-Vincentelli. 2019. An Encoder-Decoder Based Approach for Anomaly Detection with Application in Additive Manufacturing. In 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA). IEEE, 1008–1015.

[35] Jingxin Zhang, Hao Chen, Songhang Chen, and Xia Hong. 2017. An improved mixture of probabilistic PCA for nonlinear data-driven process monitoring. IEEE transactions on cybernetics 49, 1 (2017), 198–210.

[36] Zhi-Hua Zhou. 2012. Ensemble methods: foundations and algorithms. Chapman and Hall/CRC.

[37] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. (2018).
A IMPLEMENTATION DETAILS

A.1 Experiments on the Chiller Dataset

The RP-1043 chiller dataset [5] is not public but is available for purchase from ASHRAE. The 90-ton chiller studied in the RP-1043 chiller dataset is representative of chillers used in larger installations [26], and consisted of the following parts: evaporator, compressor, condenser, economizer, motor, pumps, fans, and distribution pipes etc. with multiple sensor mounted in the system. Fig. 10 depicts the cooling system with sensors mounted in both evaporation and condensing circuits.

The same sixteen features and six fault types as used in our previous work [34] were selected to train our models in the case study. We also attempted to use other sets of selected features than the aforementioned sixteen features, e.g., the features identified in Li et al.'s work [24], and we obtained similar results. Detailed descriptions of the sixteen selected features and the six fault types are given in Table 1 and Table 2. Each fault was introduced at four Severity Levels (SLs), and we put fault data of all four SLs into the fault class.

For the chiller dataset, we used the sklearn package [29] for implementing the Machine Learning (ML) models used in our experiments. A few outlier points were first removed, and the data were first standardized before they were used for training. We experimented using Decision Tree (DT), Neural Network (NN), One-Class Support Vector Machine (OC-SVM) as base learners for constructing ensembles. The base learners were all implemented by using existing modules in sklearn; see our released code for further details.

To carry out the hyperparameter search, we utilized the GridSearchCV module in sklearn to sweep over the prescribed hyperparameter space. For DT models, we swept the max_depth parameter over the range \{8, 10, 12, 15, 20\} and attempted various parameters configurations such as criterion (measuring the quality of split) and splitter (strategy used to choose the split at each node). For NN models (multilayer perceptrons), we tried several different network topologies with depth ranging from 2 to 5, various batch sizes (32, 64, 128) and optimizer settings (sgd or

---

Table 1. Descriptions of the selected variables for the chiller dataset

| Sensor    | Description                              | Unit |
|-----------|------------------------------------------|------|
| TEI       | Temperature of entering evaporator water | °F   |
| TEO       | Temperature of leaving evaporator water  | °F   |
| TCI       | Temperature of entering condenser water  | °F   |
| TCO       | Temperature of leaving condenser water   | °F   |
| Cond Tons | Calculated condenser heat rejection rate | Tons |
| Cooling Tons | Calculated city water cooling rate | Tons |
| kW        | Compressor motor power consumption       | kW   |
| FWC       | Flow rate of condenser water             | gpm  |
| FWE       | Flow rate of evaporator water            | gpm  |
| PRE       | Pressure of refrigerant in evaporator     | psig |
| PRC       | Pressure of refrigerant in condenser      | psig |
| TRC       | Subcooling temperature                    | °F   |
| T_suc     | Refrigerant suction temperature           | °F   |
| Tsh_suc   | Refrigerant suction superheat temperature| °F   |
| TR_dis    | Refrigerant discharge temperature         | °F   |
| Tsh_dis   | Refrigerant discharge superheat temperature | °F |
Table 2. The six chiller faults in the chiller dataset.

| Fault Type                      | Normal Operation | Emulation Method                                      |
|--------------------------------|------------------|-------------------------------------------------------|
| Reduced Condenser Water Flow (FWC) | 270 gpm          | Reducing water flow rate in condenser                  |
| Reduced Evaporator Water Flow (FWE) | 216 gpm          | Reducing water flow rate in evaporator                 |
| Refrigerant Leak (RL)            | 300 lb           | Reducing the refrigerant charge                        |
| Refrigerant Overcharge (RO)      | 300 lb           | Increasing the refrigerant charge                      |
| Condenser Fouling (CF)           | 164 tubes        | Plugging tubes into condenser                          |
| Non-condensables in System (NC)  | No nitrogen      | Adding Nitrogen to the refrigerant                     |

Adam). For OC-SVM models with Radial Basis Function (RBF) kernels, we conducted a grid search over parameters $\nu$ and $\gamma$ [16]. We refer interested readers to our released code base for further implementation details.

After the grid search, the top $R$ sets of hyperparameters for each model type were picked out and used for constructing the base learners for bagging ensembles. The bagging ensembles in this study were implemented using the Bagging module from sklearn, which enabled us to create bagging models with different types of base learners. The sizes of random sample subsets for training each base model can be specified through the max_samples argument.

A.2 Experiments on the Diabetic Retinopathy (DR) Dataset

We used Convolutional Neural Network (CNN) models for classifying image models in the DR dataset. The CNN models were implemented using pytorch [28]. The deep learning models used to construct our ensembles vary in their architecture, image data resolution, training set selection, number of training epochs and data augmentation strengths. Two different CNN architectures, ResNet34 [13] and VGG16 [33], were used in our experiments. We used the binary-cross-entropy loss function and the Adam [18] optimizer during training. All network parameters were initialized with the weights from pretrained models provided by the torchvision [25] package that were created for classifying objects from the ImageNet database.

Since our experiments involved scanning various $\rho$ values, to reduce the total training effort, we first trained our models with non-incipient disease data (only SL0 & SL3 & SL4) for 130 epochs, and

![Fig. 10. A schematic of the cooling system test facility and sensors mounted in the related water circuits [22].](image)
then continued to train the resulting networks with all training data (SL0 to SL4) till convergence. Most trained models reached an Area Under Curve (AUC) of above 0.98 on both the training and the validation sets. We discarded the bad performing models and put the rest into a pool. The retained models in the pool were then used as base learners for constructing ensembles. To create an ensemble model instance, we randomly picked \( K \) single learners from the pool. In our experiment, we evaluated \( K = 5 \) and \( K = 10 \), two ensemble sizes used in previous works [11, 20]. Their individual predictions were then combined and grouped for later analysis.

Data preprocessing. The image data used in our experiments were all unified into square-shaped images with resolutions 224 × 224 or 384 × 384 in our preprocessing procedure. For training each neural network model, only images of the same resolution were used. The original images came with either of the two formats as exemplified in Fig. 11. In the first format as shown in Fig. 11a, the entire fundus was visible in the image. We cropped the original image such that the fundus would tightly fitted inside the square. In the second format of input images shown in Fig. 11b, part of the fundus was not visible. We padded blank strips to make the image square-shaped and in a unified resolution; see our released code for further details.

![Fig. 11. Preprocessing the fundus image data from the Kaggle-DR dataset [6].](image)

Data augmentation. Data augmentation [30] has proved to be an important technique for training deep learning models that can prevent overfitting and can enhance model’s generalization ability. We utilized several different types of data augmentation operations at training time that are available from the torchvision package [25]. These operations included RandomResizedCrop, adjust_brightness, adjust_saturation and adjust_contrast that could randomly adjust the aspect ratio, the brightness, the saturation and the contrast respectively. The degree (strength) of data augmentation in our experiments was controlled by a multiplier \( \gamma \in \{0.1, 0.3, 0.5, 0.7\} \); see the provided code for further details.

B MODELING THE DISTRIBUTIONS OF ENSEMBLE OUTPUTS

In this paper, we assume in our theoretical analysis that the responses of each member classifier in an ensemble follow a Beta distribution \( \mathcal{B}(\alpha, \beta) \). We visualized the distributions of individual learner outputs for a select number of data points from both datasets in Figs. 12 & 13 & 14. For each model type and dataset, we randomly selected nine data points for each SL, and showed as histograms the variations of the trained models’ predictions on each data point.
Fig. 12. Histograms showing the spread of trained (with $\rho = 0.2$) DT models’ predictions on a selected numbers of data examples from the chiller dataset, and a fitted Beta distribution $\mathcal{B}(\alpha, \beta)$ for each example.

(a) SL0  
(b) SL1  
(c) SL2  
(d) SL3  
(e) SL4
Fig. 13. Histograms showing the spread of trained (with $\rho = 0.2$) NN models’ predictions on a selected numbers of data examples from the chiller dataset, and a fitted Beta distribution $\mathcal{B}(\alpha, \beta)$ for each example.
Fig. 14. Histograms showing the spread of trained (with $\rho = 0.2$) NN models’ predictions on a selected numbers of data examples from the diabetic dataset, and a fitted Beta distribution $B(\alpha, \beta)$ for each example.